Межведомственный суперкомпьютерный центр Российской академии наук –
филиал Федерального государственного учреждения
«Федеральный научный центр Научно-исследовательский институт системных исследований Российской академии наук»
Федеральное государственное учреждение «Федеральный исследовательский центр Институт прикладной математики им. М.В. Келдыша Российской академии наук»

Система управления прохождением параллельных заданий (СУППЗ)

Руководство программиста (пользователя)

Москва 2021

Аннотация

Документ содержит сведения, необходимые для программистов, использующих систему управления прохождения заданий (СУППЗ). СУППЗ предназначена для планирования и распределения параллельных заданий по ресурсам многопроцессорной вычислительной системы. При использовании СУППЗ программист выступает в качестве **пользователя** системы, поэтому в тексте документа термины «программист» и «пользователь» следует считать синонимами. При этом термин «пользователь», как правило, в документе употребляется в контексте действий программиста как пользователя операционной системы.

В разделе 1 представлены сведения о назначении и принципах функционирования СУППЗ.

Раздел 2 содержит сведения о характеристиках СУППЗ, функциональной структуре и используемых алгоритмах обработки заданий.

Раздел 3 содержит информацию о командах, с помощью которых осуществляется обращение программиста к СУППЗ.

Раздел 4 содержит информацию о входных и выходных данных, используемых программистом при работе с СУППЗ.

Раздел 5 содержит описание основных диагностических сообщений СУППЗ программисту.

Содержание

1 Назначение и условия применения СУППЗ

1.1 СУППЗ предназначена для планирования и распределения параллельных заданий по ресурсам многопроцессорной вычислительной системы.

1.2 Многопроцессорная вычислительная система, находящаяся под управлением СУППЗ, должна иметь следующий примерный состав:

- многопроцессорный вычислитель;
- управляющая ЭВМ;
- сервер доступа;
- файловый сервер (система хранения данных).

Вычислитель состоит из соединенных одной или несколькими высокоскоростными сетями вычислительных модулей (ВМ), каждый из которых уникально именован в системе.

Для примера будем считать, что условное имя управляющей ЭВМ – `head`, условное имя сервера доступа – `login`, а условные имена ВМ – `node1`, `node2`, `node3` и т.д. ВМ образуют **решающее поле** вычислителя.

1.3 Характеристиками ВМ решающего поля являются:

- уникальное сетевое имя ВМ (в примере из рисунка 1 – `node1`, `node2` и `node3`);
- число процессоров (ядер) на ВМ;
- дополнительные вычислительные ресурсы в соответствии с 1.6.

1.4 Пользовательское задание выполняется на одном или нескольких процессорах вычислителя. Задания могут запускаться и завершаться независимо друг от друга. Вычислитель делится между заданиями динамически с точностью до ВМ. В то же время, отдельный ВМ не делится между заданиями: если одно задание получило некоторый ВМ, другое задание воспользоваться им не сможет до завершения первого. При запуске задания на счет оно получает набор ВМ с произвольными именами из числа свободных. При этом другие ВМ заданию недоступны.

1.5 Каждый ВМ имеет свой текущий **статус**:

- свободен (`free`) – ВМ исправен, но в текущий момент времени не используется ни одним заданием;

- используется (`used`) – ВМ исправен, и в текущий момент времени выделен для выполнения некоторого задания;

- блокирован (`locked`) – ВМ выведен из решающего поля либо автоматически по причине обнаруженной неисправности, либо по команде оператора.

1.6 Основным вычислительным ресурсом ВМ являются процессоры (ядра). Кроме этого, часть ВМ решающего поля может содержать **дополнительные вычислительные ресурсы**, например, память увеличенного объема или операционную систему определенной версии. Дополнительные вычислительные ресурсы специфицируются системным программистом. Каждый специфицированный дополнительный ресурс имеет имя и может быть **числовым** или **строковым**. Текущую конфигурацию вычислителя можно определить согласно 3.3.6. При постановке задания в очередь (запуске) пользователь в соответствии с 3.1.1.13, 3.1.1.14, а также с 4.1.4, может указать, какие дополнительные вычислительные ресурсы требуются для его задания.

1.7 **Сервер доступа** служит для подсоединения пользователей к системе, подготовки и хранения исходных текстов программ, данных для пользовательских заданий, результатов расчетов, для трансляции параллельных программ и подготовки заданий.

1.8 Через **управляющую ЭВМ** осуществляется доступ пользователей к вычислителю, запуск (завершение, управление) на многопроцессорном вычислителе пользовательских заданий. На системах с небольшим числом ВМ функции управляющей ЭВМ может выполнять сервер доступа.

1.9 Домашние каталоги пользователей, имеют, как правило, имена `/home/имя_пользователя` и размещаются на файловом сервере. Файловая система `/home` доступна с управляющей ЭВМ и со всех вычислительных модулей решающего поля.

1.10 На вход СУППЗ поступают различные задания от различных пользователей. Каждое задание снабжается своим паспортом, который содержит требования к ресурсам и условиям обработки задания. Задание направляется в СУППЗ программистом путем выполнения на сервере доступа специальной клиентской команды в соответствии с 3.1.1 или 3.1.2.

1.11 СУППЗ обеспечивает в автоматическом режиме:

- прием входного потока заданий;

- выделение для задания требуемых ресурсов;

- при невозможности немедленного выделения требуемых ресурсов – ведение очереди заданий;

- освобождение занятых заданием ресурсов в любой момент времени.

1.12 СУППЗ ведет очередь пользовательских заданий в соответствии с принципами, изложенными в 2.2. Программист имеет возможность просмотра состояния очереди в любой момент времени в соответствии с 3.2.

1.13 Все компоненты СУППЗ функционируют под управлением ОС Linux, поэтому для работы СУППЗ программист должен обладать навыками работы с ОС Linux.

1.14 Как пользователь ОС Linux, программист обладает именем пользователя (`login`) и принадлежит к одной или нескольким группам. Как правило, первичная группа (`primary group`) связывает пользователя-программиста с его организацией, вторичные группы (`suplimentary groups`) связывают пользователя-программиста с исследовательскими проектами, в которых он принимает участие.

1.15 Принадлежность задания к тому или иному проекту определяется СУППЗ по принадлежности каталога, из которого запускается задание, к той или иной вторичной группе пользователя-программиста. Допустим, некоторый пользователь `user` имеет первичную группу `ipm`, обозначающую его принадлежность к определенной организации (в нашем примере – ИПМ им. М.В.Келдыша РАН), и принадлежит вторичным группам `project1` (связана с проектом «Проект № 1») и `project2` (связана с проектом «Проект

№ 2»). Пусть оператором или системным программистом для работы с проектами «Проект № 1» и «Проект № 2» в сетевом хранилище данных выделены каталоги `/home/project1` (принадлежит группе `project1`) и `/home/project2` (принадлежит группе `project2`). Пусть пользователю-программисту user разрешена работа в обоих указанных каталогах. Тогда, если программист `user` будет запускать задание из каталога `/home/project1`, то СУППЗ соотнесет это заданием с проектом «Проект № 1», а если из каталога `/home/project2` – то с проектом «Проект № 2».

1.16 Системным программистом (администратором) может быть запрещен запуск заданий из домашнего каталога пользователя, т.е. из каталога, принадлежащего первичной группе пользователя-программиста. В этом случае запуск заданий для программиста будет возможен только из каталогов, принадлежащих группам проектов, в которых участвует программист.

2 Характеристика СУППЗ

2.1 Общая характеристика СУППЗ

2.1.1 СУППЗ представляет собой набор следующих программных компонентов:

- сервер очередей;

- служебные серверные процессы: сервер запросов, сервер запуска, менеджер задания;

- клиентские процессы (команды программиста);

- служебные процедуры выделения и освобождения вычислительных ресурсов;

- команды и утилиты программиста, оператора и системного программиста.

2.1.2 Программист осуществляет обращение к СУППЗ путем выполнения соответствующих команд и утилит. Каждая команда или утилита, как правило, представляет собой командный файл (скрипт), из которого производится обращение к клиентской утилите СУППЗ под названием `runmvs`.

2.1.3 Клиентская утилита `runmvs` вызывается на сервере доступа и служит для связи сервера доступа с управляющей ЭВМ. Вызов утилиты `runmvs` влечет за собой соединение с управляющей ЭВМ и автоматический запуск на управляющей ЭВМ сервера запросов.

2.1.4 Сервер запросов проводит авторизацию пользователя в СУППЗ, после чего принимает от клиентской утилиты и выполняет команду программиста, возвращая в клиентскую часть результат выполнения.

2.1.5 Ядром СУППЗ является планировщик – сервер очередей `qserv`, управление которым осуществляется посредством расписания. Расписание отражает принципы построения очереди заданий СУППЗ, изложенные в 2.2. Задания в планировщик поступают через клиентскую утилиту и сервер запросов.

2.1.6 После прохождения задания через очередь, оно запускается на выполнение сервером запуска. Выполнение каждого запущенного задания

контролируется специальным процессом на управляющей ЭВМ – **менеджером** задания. Менеджер задания ведет журнал в специальном файле в соответствии с 4.2.4.

2.1.7 СУППЗ позволяет разбивать вычислитель на две и более **логических систем**. Вычислительные модули, отнесенные к той или иной логической системе, могут пересекаться, но могут быть и разными. Кроме этого, механизм логических систем позволяет работать одной управляющей ЭВМ сразу с несколькими вычислителями. Логические системы различаются по именам. Разбиение на логические системы осуществляется системным программистом. В СУППЗ всегда присутствует логическая система по умолчанию с именем `default`.

2.1.8 При постановке задания в очередь ему, помимо задаваемого пользователем символьного имени, будет присвоен уникальный номер. Полное имя задания будет складываться из символьного имени и, через точку, – уникального номера. Например, если пользователь задал для задания имя `testmod`, а СУППЗ при постановке в очередь присвоила ему номер `2`, то полное имя задания будет `testmod.2`. Кроме этого, если определена логическая система, в которой будет выполняться задача, то имя логической системы будет добавлено в начало полного имени задачи. В нашем примере, если задание `testmod.2` будет направлено в логическую систему `mpi`, то полное имя задания будет `mpi.testmod.2`. Если задание было направлено в логическую систему по умолчанию (с именем `default`), то добавление имени логической системы к имени задания производиться не будет.

2.1.9 Для каждого запущенного задания СУППЗ создает специальный **каталог ввода-вывода** задания, в котором размещаются следующие файлы:

- `output` – файл, куда перенаправляется стандартный поток вывода процессов задания;

- `errors` – файл, куда перенаправляется стандартный поток ошибок процессов задания;

- `input` – файл стандартного ввода процессов задания, создается как

символическая ссылка на файл стандарнтого ввода, только если этот файл был задан при запуске задания в соотсвествии с 3.1.1.6 или с 4.1.1.5;

- `.hosts` – файл со списком ВМ, выделенных для задания;

- `manager.log` – файл журнала процесса-менеджера задания;

- `имя_задания.sh` – командный файл сценария выполнения задания в соответствии с 4.1.1.6, файл имеет расширение `.sh`, а его имя совпадает с именем задания.

Имя каталога ввода-вывода задания совпадает с полным именем задания, создается он, как правило, в том каталоге, из которого программист запускает задание, изменить его расположение можно в соответствии с 3.1.1.5 или с 4.1.2.3.

2.1.10 При запуске или постановке в очередь задания программистом должны быть сформулированы требования к вычислительным ресурсам решающего поля. Минимальный набор требований включает **число процессоров (ядер)** и **максимальное время** выполнения задания. Если задание не завершится до истечения максимального времени выполнения, оно будет принудительно снято с выполнения, о чем появится сообщение в журнале процесса-менеджера в соответствии с 4.2.6. Полный набор требований к вычислительным ресурсам формулируется в специальном файле – **паспорте задания** в соответствии с 4.1.1.

2.1.11 Паспорт задания автоматически будет сформирован для MPI-программ при использовании команды `mpirun` в соответсвии с 3.1.1.

2.1.12 Пользовательские настройки клиентской части СУППЗ задаются в пользовательском конфигурационном файле с именем `.crunmvs`, расположенном в домашнем каталоге пользователя. Формат пользовательского конфигурационного файла рассмотрен в 4.1.2.

2.2 Планирование очереди в СУППЗ

2.2.1 При планировании все задания делятся на три категории – **отладочные** (короткие по времени задания, использующие малое число процессоров), **пакетные** (продолжительные по времени задания с

произвольным числом процессоров) и **фоновые** (очень продолжительные задания, которые могут прерываться системой).

2.2.2 Для фонового задания программист-пользователь явно указывает **квант** – минимальное время выполнения фонового задания. СУППЗ гарантирует, что если фоновое задание было запущено, то ему для выполнения будет предоставлено время, не меньшее указанного кванта. Если по истечении кванта будут обнаружены задания, претендующие на занятые фоновым заданием ресурсы, фоновое задание будет снято с выполнения и заново поставлено в очередь. При этом общее время счета фонового задания будет уменьшено на число минут, прошедших с его последнего запуска.

2.2.3 **Расписание** представляет собой последовательность сменяющих друг друга **режимов планирования**. Планирование очереди в каждый момент времени производится в соответствии с параметрами текущего режима планирования. В системе может быть несколько расписаний, переключение между которыми осуществляется по директиве оператора или системного программиста, а переключение режимов внутри каждого расписания – автоматически.

2.2.4 **Режим планирования** определяется следующими параметрами.

2.2.4.1 **Дата и время включения** режима, определяющие время, начиная с которого параметры режима вступают в силу. Параметры режима действуют вплоть до включения следующего режима.

2.2.4.2 **Шкала доступа к режиму** представляет собой двоичное 7-битовое беззнаковое число. Для каждого пользователя в СУППЗ также определяется аналогичная шкала доступа к режимам. При планировании заданий серевер очередей производит логическое побитовое умножение шкалы доступа к текущему режиму и шкалы доступа пользователя к режимам. Если в результате получается не 0, то пользователю в текущем режиме разрешается запуск и выполнение заданий, если 0 – выполнение заданий в текущем режиме запрещается.

2.2.4.3 **Общее число планируемых процессоров** – обычно задается в соответствии с суммарным числом процессорных ядер на всех ВМ решающего поля.

2.2.4.4 **Максимальное время** $T_{отл}$ и **число процессоров** $P_{отл}$, отведенные для отладочных заданий. Задания, требующие для исполнения времени, не превышающего $T_{отл}$ и числа процессоров, не превышающего $P_{отл}$, автоматически будут отнесены к категории отладочных заданий. Все пакетные задачи в сумме не могут занимать процессоров больше, чем разность между общим числом процессоров и значением $P_{отл}$ на время, большее $T_{отл}$. Фактически процессоры числом $P_{отл}$ в текущем режиме планирования будут использоваться только для счета отладочных заданий, другими словами «резервируются» для отладочных заданий. Следует пояснить, что «резервирование» не означает выделение конкретных процессоров или ВМ под отладочные задания. СУППЗ гарантирует, что $P_{отл}$ процессоров не будет использовано для пакетных заданий на время, не превышающее $T_{отл}$, а какие конкретно ВМ будут «зарезервированы», зависит от текущей ситуации. В дневных режимах планирование разумное заданий величин $T_{отл}$ и $P_{отл}$ позволяет обеспечить постоянное наличие свободных процессоров для отладочных заданий.

2.2.4.5 **Максимальное время**, отведенное для пакетных заданий – задается системным программистом или оператором как максимально возможное время исполнения задания. Задания, поступившие в СУППЗ со временем обработки, превышающим максимальное время пакетного задания, не смогут быть обработаны в текущем режиме планирования и будут ожидать в очереди смены режима или расписания.

2.2.4.6 Шкала приоритетов пользователей. Задания планируются СУППЗ согласно приоритетам пользователей. Приоритеты определяются по соответствующей шкале и напрямую зависят от суммарного времени счета пользователя за **учетный период**, который задается одинаковым для всех

режимов текущего расписания. Размер учетного периода задается системным программистом при составлении расписания очереди.

Например, если шкала имеет следующий вид

```
(120,300,600,1200,0)
```

то это означает, что наивысшим приоритетом (*очередь 0*) будут обладать задания пользователей, которые за учетный период считали менее 120 минут, чуть меньшим (*очередь 1*) приоритетом – тех, кто считал менее 300 минут, еще меньшим (*очередь 2*) – тех, кто считал менее 600 минут и т.д. Низшим для текущего режима (*очередь 4*) приоритетом будут обладать задания пользователей, считавших более 1200 минут.

2.2.4.7 При суммировании времени заданий пользователя за учетный период, время каждого задания умножается на специальный коэффициент – **цену задания**. Цена задания определяется системным программистом для каждого пользователя при составлении расписания.

2.2.4.8 При вычислении приоритета задания учитывается заказанное время его выполнения, которое прибавляется к суммарному времени заданий пользователя за учетный период.

2.2.5 Планировщик пытается выделить ресурсы из числа свободных сначала для заданий из очереди 0, потом — из очереди 1 и т.д. Внутри одной очереди ресурсы выделяются в порядке (от меньшего к большему) номеров в списке заданий. Номер очереди и номер задания в очереди образуют **составной индикатор порядка выделения ресурсов**. Если свободных ресурсов для задания нет, определяется момент времени, когда нужные ресурсы освободятся, и устанавливается время запуска задания. Никакое менее приоритетное задание не может занять ресурсы так, чтобы это отодвинуло запуск более приоритетного задания. При отсутствии конфликта по ресурсам менее приоритетное задание может стартовать раньше более приоритетного.

Планировщик может запускать задания, стоящие в очереди позже, если это не повлияет на время старта задания, стоящего в очереди ранее (применяется т.н. алгоритм «обратного заполнения»). Такое возможно,

например, в случае, если для задания А, стоящего в очереди ранее, недостаточно ресурсов, и при этом задание Б, стоящее в очереди позже, успеет завершиться до момента, когда освободится достаточное количество ресурсов для запуска задания А.

2.2.6 При постановке задания в очередь пользователь имеет возможность указать время предупреждения задания о его предстоящем завершении. Время это может быть любым, но не больше установленного системным программистом или оператором предела. Далее будем называть это время **дополнительным временем**.

2.2.6.1 Задания тех пользователей, которые не заказали дополнительное время, планируются обычным образом. Если дополнительное время заказано, то оно будет прибавляться к основному времени (для отладочных и пакетных заданий), либо к кванту (для фоновых заданий).

2.2.6.2 Для отслеживания времени снятия задачи со счета программист имеет возможность в соответствии с 3.1.1.11 или с 4.1.1.4 дать СУППЗ указание о рассылке определенного сигнала всем процессам задания. СУППЗ рассылает сигнал о предстоящем снятии задания с выполнения только в том случае, когда программист явно указал при запуске задания дополнительное время и сигнал в соответствии с 3.1.1.10, 3.1.1.11 или с 4.1.1.4. Программист должен самостоятельно определить обработчик сигнала в своей программе.

2.2.6.3 Программист должен обеспечить выполнение заданием всех необходимых действий для завершения при получении извещения о предстоящем снятии с выполнения. По истечении заданного дополнительного времени задание будет безусловно снято с выполнения.

2.2.6.4 Системой гарантируется, что отладочные и пакетные задания при запуске сначала получат для счета все заказанное время, по истечении которого задание будет извещено о снятии с выполнения, и ему будет выделено дополнительное время для завершения. Для фоновых заданий при запуске гарантируется получение, как минимум, одного указанного кванта. Если по истечении кванта планировщик принимает решение о снятии задания, то

сначала процессы задания предупреждаются об этом путем рассылки заданного программистом сигнала в соответствии с 3.1.1.11 или с 4.1.1.4, а затем выделяется дополнительное время для завершения.

3 Обращение к СУППЗ

3.1 Запуск и завершение заданий

3.1.1 Пусть имеется исполнительный модуль параллельной MPI-программы с именем `MPI_prog`, которую необходимо запустить с параметрами `MPI_prog_args` на решающем поле с числом процессоров (ядер) `nproc`. Запуск задания с подобной MPI-программой осуществляется командой

```
mpirun -np <nproc> [mpirun_args] <MPI_prog> [MPI_prog_args]
```

где `nproc` – число требуемых для задания процессоров (ядер);

`mpirun_args` – параметры команды `mpirun`;

`MPI_prog` – имя исполняемого модуля MPI-программы;

`MPI_prog_args` – параметры MPI-программы.

Команда `mpirun` автоматически формирует паспорт задания и направляет его в очередь СУППЗ. Завершение команды `mpirun` не означает завершение или запуск задания (за исключением случая, рассмотренного в 3.1.1.4), а лишь завершение постановки задания в очередь. При успешном завершении команда `mpirun` возвращает 0, иначе – отличное от 0 значение с выдачей диагностического сообщения в сандартный поток ошибок. Исключение составляет случай, рассмотренный в 3.1.1.17.

Все параметры команды `mpirun` являются ключевыми, т.е. предваряются ключевым словом со знаком `-`.

3.1.1.1 Параметр

```
-maxtime <максимальное_время>
```

задает максимальное время выполнения задания в минутах. После истечения этого времени задание будет принудительно завершено. Сообщение о принудительном завершении задания будет выведено в журнал процесса-менеджера задания в соответствии с 4.2.6. Указанное в параметре `-maxtime` время будет участвовать в расчете приоритета задания в соответствии с 2.2.4.6 – 2.2.4.8. Если параметр `-maxtime` не будет задан, максимальное время

выполнения задания будет взято из пользовательского конфигурационного файла в соответсвии с 4.1.2.3.

3.1.1.2 Параметр

`-quantum <значение_кванта_времени>`

задает время (в минутах) кванта для фонового задания. Если параметр `-quantum` не будет задан, значение кванта для фонового задания будет взято из пользовательского конфигурационного файла в соответсвии с 4.1.2.3.

3.1.1.3 Параметр

`-restart`

указывает, что после своего завершения задание будет вновь поставлено в очередь. Для удаления из очереди такого задания необходимо воспользоваться командой `mqdel` в соответствии с 3.1.4, а для завершения выполняющегося задания – командами `mkill` или `mterm` в соответствии с 3.1.5. Если параметр не задан, по умолчанию будет применяться значение параметра `restart` секции `[Mpirun]` пользовательского конфигурационного файла в соответствии с 4.1.2.3.

3.1.1.4 Параметр

`-wait`

указывает, что в случае успешного запуска задания (либо постановки в очередь) команда `mpirun` будет ожидать его завершения. Для отладочных и пакетных заданий под завершением здесь понимается обычное завершение или принудительное снятие с выполнения (по команде программиста или оператора, либо по истечении времени). Для фонового задания – завершение последнего кванта, либо снятие с выполнения командой `mterm` в соотвествии с 3.1.5.

3.1.1.5 Параметр

`-stdiodir <имя_каталога>`

задает имя текущего каталога задания, в котором в соответствии с 2.1.9 будет созаваться каталог ввода-вывода задания. Если параметр не задан, его значение

будет взято из пользовательского конфигурационного файла в соответсвии с 4.1.2.3.

3.1.1.6 Параметр

`-stdin <имя_файла>`

задает имя файла, на который будет перенаправлен поток стандартного ввода процессов задания. На указанный файл в каталоге ввода-вывода задания автоматически будет создана символическая ссылка с именем `input`. Если параметр не указан, стандартный ввод для процессов задания будет недоступен и не должен использоваться в параллельной программе.

3.1.1.7 Параметр

`-stdout <имя_файла>`

задает имя файла, в который будет перенаправлен стандартный поток вывода процессов задания.

3.1.1.8 Параметр

`-stderr <имя_файла>`

задает имя файла, в который будет перенаправлен стандартный поток ошибок процессов задания.

3.1.1.9 Параметр

`-interactive`

делает задачу интерактивной в соответствии с 3.4.3, а также отменяет действия параметров `-stdin`, `-stdout`, `-stderr`.

3.1.1.10 Параметр

`-termtime <дополнительное_время>`

задает дополнительное время для завершения задания в соответствии с 2.2.6. Дополнителное время задается в минутах. Если параметр `-termtime` не будет задан, значение дополнительного времени для завершения задания будет взято из пользовательского конфигурационного файла в соответсвии с 4.1.2.3.

3.1.1.11 Параметр

`-termsignal <сигнал_для_завершения>`

имеет действие только, если задано дополнительное время для завершения задания в соответствии с 3.1.1.10. Параметр `-termsignal` задает сигнал, который будет разослан всем процессам задания в качестве предупреждения о предстоящем завершении в соответствии с 2.2.6.2. Формат задания сигнала должен соответствовать команде ОС Linux `kill`. Если параметр `-termsignal` не будет задан, значение сигнала будет взято из пользовательского конфигурационного файла в соответсвии с 4.1.2.3.

3.1.1.12 Параметр

`-machinefile <имя_файла_шаблона>`

задает имя файла-шаблона, позволяющего изменять порядок распределения MPI-процессов по вычислительным модулям в соответствии с 4.1.3.

3.1.1.13 Параметр

`-resource "имя_ресурса=значение_ресурса"`

задает требования к дополнительному ресурсу с именем `имя_ресурса` и значением `значение_ресурса` для всех ВМ, выделяемых заданию. Имена и значения дополнительных ресурсов определяются в соответствии с 3.3.6. Задание будет назначено только на те ВМ, которые обладают указанным в параметре `-resource` ресурсом с именем `имя_ресурса`. При этом числовые ресурсы подбираются системой так, чтобы их значения на выделяемых ВМ были не меньше заданного в параметре `-resource` значения `значение_ресурса`. Строковые ресурсы подбираются системой так, чтобы их значения на выделяемых ВМ точно соответствовали заданному в параметре `-resource` значению `значение_ресурса`.

Пусть конфигурация дополнительных ресурсов соответствует примеру из 3.3.6. Допустим, для программы с именем `cpi`, рассчитанной на выполнение на 10 процессорах, требуется, чтобы на всех выделенных ВМ было не менее 16 Гб оперативной памяти. Команда `mpirun` в этом случае должна выглядеть следующим образом

```
mpirun -np 10 -resource "mem=16" cpi
```

В результате для задания будут выделены модули `node02` и `node03`, т.к. на каждом имеется не менее 16 Гб памяти. Если же задать значение ресурса 32 Гб, то будет выдана ошибка, т.к. задаче требуется минимум 2 модуля (10 процессоров), но в системе есть только один (`node02`), обладающий необходимым ресурсом.

Параметр `-resource` может при необходимости повторяться в команде `mpirun` для возможности указания нескольких вычислительных ресурсов, например

```
mpirun -np 8 -resource "mem=16" -resource "OS=Windows" cpi
```

В этом случае для задачи будет выделен модуль `node03` (16 Гб оперативной памяти и наличие ОС Windows).

3.1.1.14 Параметр

```
-resourcefile  <имя_файла_спецификации_ресурсов>
```

задает файл-спецификацию дополнительных ресурсов для задания в соответствии с 4.1.4.

3.1.1.15 Параметр

```
-host  <имя_управляющей_ЭВМ>
```

задает имя управляющей ЭВМ, куда необходимо направить задание. Если параметр не задан, имя управляющей ЭВМ будет определено из пользовательского конфигурационного файла в соответствии с 4.1.2.2.

3.1.1.16 Параметр

```
-s  <имя_логической_системы>
```

задает имя логической системы в соответствии с 2.1.7. Если параметр не задан, будет выбрана логическая система по умолчанию.

3.1.1.17 Параметр

```
-retcode
```

предписывает команде `mpirun` в случае удачного завершения возвращать не 0, а номер запущенного (поставленного в очередь) задания (действительно для номеров меньше 128). Возвращаемый номер определяется в соответствии с

2.1.8. В случае возникновения ошибки при заданном параметре `-retcode` команда `mpirun` вернет значение от 128 до 255.

3.1.1.18 Параметр

`-ppn <число_процессов_на_ВМ>`

предписывает изменить число процессов на вычислительном модуле в соответствии с указанной в параметре величиной. По умолчанию число процессов на ВМ равно числу процессоров (ядер), параметр `-ppn` позволяет изменить это число.

Допустим, число процессоров (ядер) на каждом ВМ равно 8. Тогда по команде

```
mpirun -np 16 myprog
```

заданию `myprog` будет выделено 2 вычислительных модуля, и на каждом будет запущено по 8 процессов. Используя параметр `-ppn`, можно управлять числом процессов. Например, по команде

```
mpirun -np 16 -ppn 4 myprog
```

заданию `myprog` будет выделено 4 вычислительных модуля, и на каждом будет запущено по 4 процесса. Кроме этого управлять размещением процессов на ВМ можно с помощью файла-шаблона в соответствии с 4.1.3.

3.1.1.19 Параметр

`-passport`

позволит сохранить сформированный командой `mpirun` паспорт задания в файле с раширением `.img`, имя которого будет совпадать с именем исполняемого файла. Например, если команда `mpirun` выглядит следующим образом

```
mpirun -np 16 -passport myprog
```

то в текущем каталоге после выполнения команды останется файл `myprog,img`, содержащий паспорт задания в формате, рассмотренным в п. 4.1.1.

3.1.2 Запуск произвольного командного файла осуществляется командой

```
mbatch -np <nproc> [mpirun_args] <script> [script_args]
```

где nproc – число требуемых для задания процессоров (ядер);

mpirun_args – параметры, аналогичные параметрам команды mpirun в соответствии с 3.1.1, за исключением опций, рассмотренных ниже;

script – имя командного файла (сценария, скрипта);

script_args – параметры командного файла (сценария, скрипта).

Команда mbatch автоматически формирует паспорт задания и направляет его в очередь СУППЗ. Завершение команды mbathc не означает завершение или запуск задания (за исключением случая, рассмотренного в 3.1.1.4), а лишь завершение постановки задания в очередь. При успешном завершении команда mbatch возвращает 0, иначе – отличное от 0 значение с выдачей диагностического сообщения в сандартный поток ошибок. Исключение составляет случай, рассмотренный в 3.1.1.17.

Опции команды mbatch аналогичны опциям команды mpirun, за следующими исключениями. Команда mbatch игнорирует опции и ключи –ppn и –machinefile.

3.1.3 Постановка в очередь (запуск) задания с произвольным паспортом осуществляется командой

```
mrunf [-s <имя_системы>] [-fo] <паспорт>
```

где паспорт – имя файла-паспорта задания, сформированного в соотвествии с 4.1.1

Ключ -s задает имя системы в соответствии с 2.1.7, в которую необходимо направить задание. Если задан параметр –fo, то при успешном запуске (постановке в очередь) задания команда mrunf выведет полное имя задания в соответствии с 2.1.8 в стандартный поток вывода.

3.1.4 Удаление поставленного в очередь задания осуществляется командой

```
mqdel [-s <имя_системы>] <имя_задания>
```

где имя_задания – полное имя задания в соотвествии с 2.1.8. Ключ -s задает имя системы в соответствии с 2.1.7, из которой необходимо удалить задание.

Команда mqdel применима лишь к тем заданиям, которые ожидают выполнения в очереди.

Удаление всех поставленных в очередь заданий осуществляется командой

```
mqdel '*'
```

3.1.5 Завершение выполняющегося задания осуществляется одной из двух команд

```
mkill [-s <имя_системы>] [имя_задания]
mterm [-s <имя_системы>] [имя_задания]
```

где имя_задания – полное имя задания в соотвествии с 2.1.8. Ключ -s задает имя системы в соответствии с 2.1.7, в которой необходимо завершить задание.

Команды mkill и mterm применимы только к выполняющимся заданиям.

Завершение всех выполняющихся заданий осуществляется одной из двух команд

```
mkill '*'
mterm '*'
```

Если задание фоновое, по команде mkill оно завершит очередной квант и будет заново поставлено в очередь. Команда mterm завершает задания без возврата в очередь, т.е. производит полное завершение задания.

При отсутствии параметра в командах mkill и mterm пользователю будет выдан список всех выполняющихся заданий и предложено ввести номер (по списку) задания, которое нужно завершить. Перед завершением задания будет задан вопрос о полном завершении.

Следует отметить, что команды mkill и mterm лишь сообщают СУППЗ о необходимости завершить задания, но реально не дожидаются его завершения. После успешного выполнения команд mkill и mterm задание может находиться в системе довольно продолжительное время, которое нужно системе для корректного освобождения вычислительных ресурсов. Для

ожидания завершения задания следует использовать команду mcancel в соответствии с 3.1.6.

3.1.6 Завершение задания, независимо от того, находится оно в очереди или выполняется, осуществляется командой

```
mcancel [-s <имя_логической_системы>] [имя_задания]
```

где имя_задания – полное имя задания в соотвествии с 2.1.8. Ключ -s задает имя системы в соответствии с 2.1.7, из которой необходимо удалить задание.

Команда mcancel не закончит своего выполнения до тех пор, пока задание не покинет систему, т.е. команда mcancel, в отличие от команд mkill и mterm, рассмотренных в 3.1.5, ожидает завершения задания.

3.1.7 Ожидание завершения задания осуществляется командой

```
mwait [-s <имя_логической_системы>] [имя_задания]
```

где имя_задания – полное имя задания в соотвествии с 2.1.8. Ключ -s задает имя системы в соответствии с 2.1.7, в которой необходимо ожидать завершение задания.

Команда mwait будет ожидать («подвиснет») до тех пор, пока стоящее в очереди или запущенное задание не закончит выполнение и не покинет систему. Для отладочных и пакетных заданий этот момент наступает во время завершения. Для фонового задания – во время завершения последнего кванта либо снятия с выполнения по команде mterm в соответствии с 3.1.5. Кроме использования команды mwait, дождаться завершения задания можно, задав ключ -wait в команде mpirun в соответствии с 3.1.1.4.

3.2 Просмотр очереди СУППЗ

3.2.1 Просмотр очереди СУППЗ осуществляется командой

```
mqinfo [-s <имя_логической_системы>]
```

где ключ -s задает имя системы в соответствии с 2.1.7.

3.2.2 Примерный вывод команды просмотра очереди СУППЗ приводится на рисунке 1.

```
Current time:  Mon Mar 22 13:10:07 2014
Queue state at Mon Mar 22 13:06:09 2014
     Current schedule: 2
     Accumulation period: 168 hours

--- 22.03.14 07:30: R=1 total-30 debug-0/0, packet-300, priority scale-(10000)
stok3.2:lara     0.1  0*  12  150/150      16    >:03.22.14 18:03
intro.1:kira     0.2  0*  6   60/60+10         6    >:03.22.14 18:05
-- queue --
color.2:vera     0.3  0*  12  100/100      b~~~    <:03.22.02 14:33
--- 23.03.14 07:30: R=1 total-30 debug-0/0, packet-300, priority scale-(10000)
--- 24.03.14 07:30: R=1 total-30 debug-0/0, packet-300, priority scale-(10000)

Free: 0 proc   Available: 18  Locked: 12

   lara: Inf:3 R=1 run=1 wait=0/5 all=0/7   *1.00 Sr/Sf=10/26
   kira: Inf:3 R=1 run=1 wait=0/5 all=0/7   *1.00 Sr/Sf=10/26
   vera: Inf:3 R=1 run=0 wait=0/5 all=1/7   *1.00 Sr/Sf=10/26
   u1310: Block:RQ Inf:2 R=1 run=0 wait=0/5 all=0/7   *1.00 Sr/Sf=2/2
::THE_OTHERS:: Inf:3 R=1 run=0 wait=0/5 all=0/10   *1.00 Sr/Sf=0/0
   runmvs: Inf:3 R=2 run=0 wait=0/5 all=0/50   *1.00 Sr/Sf=0/0
   natasol: Inf:3 R=5 run=0 wait=0/5 all=0/50   *0.01 Sr/Sf=0/0
```

Рисунок 1 – Примерный вывод команды просмотра очереди СУППЗ

3.2.3 В заголовке выдачи сообщается, когда было в последний раз перезаписано состояние очереди, а также (`Current schedule`) – номер текущего расписания. Если очередь блокирована (заморожена), то это будет отображено словом `Hold` после номера расписания. Кроме того, в соответствии с 2.2.4.6 выдается учетный период (`Accumulation period`), за который производится суммирование времени счета завершившихся заданий одного пользователя.

3.2.4 Информация о режимах планирования начинается с «---» и содержит:

- дату и время включения режима в соответствии с 2.2.4.1;

- после `R=` – шкалу доступа к режиму в соответствии с 2.2.4.2;

- после `total` – общее число планируемых процессоров в соответствии с 2.2.4.3;

- после `debug` – количество процессоров, отведенное под отладочные задания/максимальное время (в минутах) для отладочных заданий в соответствии с 2.2.4.4;

- после `packet` – максимальное время (в минутах) для пакетных заданий

в соответствии с 2.2.4.5;

- после `priority scale` – шкала приоритетов в соответствии с 2.2.4.6.

3.2.5 Информация об обрабатывающихся заданиях содержит:

- имя задания в соответствии с 2.1.8;

- имя пользователя – владельца задания;

- составной индикатор порядка выделения ресурсов в соответствии с 2.2.5;

- количество оставшихся повторов счета, всегда 0;

- количество процессоров (ядер), занимаемых заданием;

- остаток времени обработки в минутах;

- квант счета в минутах, для отладочных и пакетных заданий всегда совпадает с временем обработки;

- после + – дополнительное время в соответствии с 2.2.6;

- количество минут до предполагаемого завершения задания;

- предполагаемое время завершения (после `>:`).

В примере на рисунке 1 обрабатываются два задания: `stok3.2` пользователя `lara` и `intro.1`. пользователя `kira`. Задание `stok3.2` выполняется на 12 процессорах с максимальным временем 150 минут, а задание `intro.1` – на 6 процессорах с максимальным временем 60 минут. Заданию `stok3.2` осталось обрабатываться 16 минут, а заданию `intro.1` – 6 минут. Для задания `intro.1` задано дополнительное время 10 минут.

3.2.6 Запуск и завершение задания отмечаются в выдаче ключевыми словами `starting` и `finishing` соответственно, например:

```
img1.3 : guest      6.3  0*  15  150/15+1      starting
img1.2 : guest      5.2  0*  15  150/150       finishing
```

3.2.7 При наличии очереди заданий в выдаче будет присутствовать фраза `-- queue --`, после которой будет располагаться список заданий в очереди. Информация о стоящих в очереди заданиях отличается от информации об обрабатывающихся заданиях тем, что вместо количества минут до завершения

выдается код (причины) ожидания и количество минут до предполагаемого старта обработки (разделенные символом ~), а вместо времени завершения – время постановки в очередь (после <:).

3.2.8 Коды причин ожидания могут быть следующие:

- A – задание заблокировано оператором;

- H – очередь в целом заблокирована оператором;

- O – рассматриваемых планировщиком режимов не хватает, чтобы сделать вывод о возможности или невозможности обработки задания в одном из них;

- P – в текущем расписании для данного пользователя запрещено запускать задания с таким числом процессоров;

- R – задание не может быть обработано ни в одном из рассматриваемых планировщиком режимов;

- e – старт отложен из-за несущественной ошибки при запуске;

- b – занятость процессоров обрабатывающимися заданиями;

- m – в одном из режимов пользователю запрещено запускать задания;

- p – при запуске задание отодвинуло бы старт более приоритетного задания.

3.2.9 В примере на рисунке 1 в очереди находится задание color.2 пользователя vera, предназначенное для выполнения на 12 процессорах с максимальным временем 100 минут. Код причины ожидания b~~ говорит о занятости процессоров обрабатывающимися заданиями. В то же время из содержимого строки вывода

```
Free: 0 proc   Available: 18  Locked: 12
```

можно определить, что свободных (Free) процессоров на текущий момент нет, доступных (Available) процессоров – 18 (все они заняты), заблокировано (Locked) 12 процессоров.

3.2.10 Последние семь строк в выдаче на рисунке 1 содержат текущую статистику по пользователям `lara`, `kira`, `vera`, `u1310`, `runmvs` и `natasol`. Информация о пользователях содержит:

- имя пользователя;

- информация о запрете (`Block:`) счета (`Q`) или запуска (`R`) заданий (если он установлен системным программистом);

- степень подробности (от 1 до 3) выдачи информации (`Inf:`) о заданиях других пользователей:

  а) степень подробности 1 – пользователю выдается информация только о его собственных заданиях;

  б) степень подробности 2 – пользователю не выдается информация о заданиях других пользователей и информация о пользователях, которые в учетный период не считали и не имеют заданий в очереди;

  в) степень подробности 3 – пользователю выдается вся информация о всех заданиях всех пользователей и вся информация о всех пользователях;

- шкала доступа к режимам (`R=`) в соответствии с 2.2.4.2;

- после `run` – количество выполняющихся заданий;

- после `wait` – количество ждущих заданий/ограничение на количество ждущих заданий;

- после `all` – суммарное количество заданий в очереди/ограничение на суммарное количество заданий в очереди;

- цена задания в соответствии с 2.2.4.7;

- после `Sr/Sf` – суммарная стоимость выполняющихся заданий/суммарная стоимость завершившихся заданий в соответствии с 2.2.4.6.

3.2.10.1 Пользователи при выдаче информации разделяются на **обычных**, для которых применяются типичные настройки в расписании, и **специальных**. Настройки специальных пользователей персонально задаются системным программистом при составлении расписания.

3.2.10.2 На рисунке 1 показано, что пользователю u1310 запрещено ставить задания в очередь, его стоящие в очереди задания заморожены. Всем пользователям информация выдается со степенью подробности 3, а пользователю u1310 – со степенью подробности 2. В соответствии с 2.2.4.2 всем пользователям, за исключением runmvs, шкала доступа к режимам позволяет запускать задания во всех приведенных на рисунке 1 режимах планирования, т.к. только для пользователя runmvs побитовое умножение его шкалы доступа (2) и шкал доступа к режимам (1) даст в результате 0.

3.2.10.3 Специальное имя пользователя ::THE_OTHERS:: зарезервировано для выдачи информации о типичных настройках обычных пользователей. После информации о настройках ::THE_OTHERS:: будут выданы настройки для специально выделенных пользователей, но только для тех, кто не имеет на текущий момент ни заданий в очереди, ни завершившихся заданий за учетный период. Информация о настройках специальных пользователей, имеющих задания в очереди или выполнявших задания за учетный период вместе с информацией о текущих значениях до строки с ::THE_OTHERS::.

3.2.11 Если очередь блокирована (заморожена), то это отображается словом Hold, как показано на рисунке 2.

```
Queue state at Wed Mar 22 12:40:07 2011
     Current schedule: 2      Hold
Accumulation period: 168 hours

--- 22.03.11 07:30: R=3 total-32 debug-8/5, packet-300, priority scale-(120 300)
--- 22.03.11 19:00: R=2 total-32 debug- 0/0, packet-600, priority scale-(10000)
    tnet.7 : ant          1*   10  15/15      H~~~   <:Mon Mar 22 13:31:09 2011
    tnet.3 : ant          1*   24  500/500  H~~~   <:Mon Mar 22 20:31:09 2011
    tnet.1 : ant          1*   1  15/15      H~~~   <:Mon Mar 20 20:31:09 2011
.............................
```

Рисунок 2 – Просмотр состояния заблокированной очереди

3.3 Получение информации о заданиях и ресурсах СУППЗ

3.3.1 Информацию о направленных в СУППЗ заданиях пользователя можно получить с помощью команды

```
mps [-s <имя_логической_системы>] [имя_задания]
```

где имя_задания – полное имя задания в соответствии с 2.1.8, ключ -s задает имя системы в соответствии с 2.1.7.

3.3.1.1 При отсутствии параметра в стандартный поток вывода будет выдан список всех выполняющихся или находящихся в очереди заданий пользователя. Если задание находится в очереди, это будет отмечено словом queued рядом с именем задания.

3.3.1.2 При задании параметра имя_задания команда mps выведет в стандартный поток вывода подробную информацию о выполняющемся или завершенном задании. Примерный формат выводимой информации о выполняющемся задании следующий:

```
Job "mpi.8" started 05.11.14 01:11:42
It's settings are:
Job directory: /home/ivanov/mpi/mpi.8
Pid of manager: 17415
CPU count: 16

        NODE                    CPU
        ------                  ---
1.      node849                 8
2.      node909                 8
```

Первая строка вывода

```
Job "mpi.8" started 05.11.14 01:11:42
```

сообщает о времени запуска (05.11.14 01:11:42) задания с полным именем (в соответствии с 2.1.8) mpi.8, после чего приводятся характеристики задания. После фразы Job directory: следует полный путь к каталогу ввода-вывода задания (в нашем примере /home/ivanov/mpi/mpi.8) в соответствии с 2.1.9. После фразы Pid of manager: следует идентификатор процесса-менеджера задания (17415), а после фразы CPU count: – число используемых заданием процессоров (ядер).

После этого выводится таблица со столбцами NODE и CPU, в которых приводятся имена выделенных для задания ВМ и число используемых

процессоров (ядер) на каждом ВМ соответственно. В нашем примере для задания `mpitask.8` было выделено 2 ВМ с именами `node849` и `node909` по 8 процессоров (ядер) на каждом.

3.3.1.3 Выводимая информация о завершенном задании соответствует 3.3.1.2, за исключением первой строки выдачи, формат которой следующий:

```
Job "mpi.8" started 05.11.14 01:11:42, done 05.11.14 02:10:02
```

После слова `started` выводится время запуска задания (`05.11.14 01:11:42`), после слова `done` – время завершения задания (`05.11.14 02:10:02`).

3.3.2 Получение полного имени (в соответствии с 2.1.8) последнего поставленного в очередь задания пользователя осуществляется командой

```
mlt [-s <имя_логической_системы>]
```

Ключ `-s` задает имя системы в соответствии с 2.1.7. При успешном выполнении команда `mlt` выведет в стандартный поток вывода полное имя (в соответствии с 2.1.8) последнего поставленного в очередь задания пользователя. Команда `mlt` выдаст имя задания, даже если оно на момент выполнения команды уже было запущено, завершилось или было удалено из очереди.

3.3.3 Определение текущего статуса задания осуществляется командой

```
mqtest [-s <имя_системы>] [-rcode] <имя_задания>
```

где `имя_задания` – полное имя задания в соответствии с 2.1.8, а ключ `-s` задает имя системы в соответствии с 2.1.7.

Команда `mqtest` выводит в стандартный поток вывода информацию о текущем статусе задания с именем `имя_задания`. Статус задания может принимать следующие значения:

- `queued` – задание ожидает в очереди;
- `started` – задание выполняется;
- `done` – задание завершено.

При задании параметра `-rcode` команда `mqtest` будет возвращать специальные коды возврата:

- 1 – задание ожидает в очереди;

- 0 – задание выполняется;

- 2 – задание завершено;

- от 128 до 255 – произошла ошибка во время выполнения команды `mqtest`.

3.3.4 Получение числа свободных процессоров (ядер) решающего поля осуществляется командой

```
mfree
```

3.3.5 Получение общего числа процессоров (ядер) решающего поля осуществляется командой

```
mproc
```

3.3.6 Получение подробной информации о вычислительных ресурсах решающего поля осуществляется командой

```
mninfo [-s <имя_системы>]
```

Ключ `-s` задает имя системы в соответствии с 2.1.7. Команда `mninfo` выдает в стандартный поток вывода информацию о ресурсах в соответствии с 1.6. Рассмотрим формат выдачи команды `mninfo` на следующем примере:

```
node1: cpu:8
node2: cpu:8 (mem=32.00, OS="Linux RedHat")
node3: cpu:8 (mem=16.00, OS="Windows")
node4: cpu:4 (gpu=2, gpu_type="NVidia Tesla")
```

В приведенном примере решающее поле вычислителя составляют четыре ВМ с именами `node1-node4`. Все ВМ, кроме `node4`, имеют по 8 процессорных ядер (число после ключевого слова `cpu`). ВМ `node4` имеет 4 процессорных ядра. Если для ВМ специфицированы дополнительные ресурсы, то в выдаче их список указывается в скобках. Ресурсы в списке отделяются друг от друга запятыми и имеют вид

```
<имя_ресурса> = <значение_ресурса>
```

Имя, тип и значение ресурса задаются системным программистом. Если ресурс строковый, то его значение заключается в кавычки. В нашем примере дополнительными ресурсами обладают модули node2, node3 и node4. Числовой ресурс mem (оперативная память) для node2 имеет значение 32.00 (Гб), а для node3 – 16 (Гб). Строковый ресурс OS (операционная система) для node2 имеет значение Linux RedHat, а для node3 – Windows. ВМ node4 обладает двумя графическими ускорителями (числовой ресурс gpu) типа NVidia Tesla (строковый ресурс gpu_type). Конкретный смысл имен и значений ресурсов определяется системным программистом СУППЗ.

3.3.7 Информацию о потребленных ресурсах можно получить с помощью команды

```
mstat [-u] [-p <project_id>] [-s <system>] [-fo]
      <begin_date> <end_date>
```

где

system – имя системы в соответствии с 2.1.7;

begin_date – с какой даты выдавать статистику, формат даты YYYY-MM-DD;

end_date – до какой даты выдавать статистику. Формат даты YYYY-MM-DD;

-u – ключ выдачи статистики пользователя;

project_id – идентификатор проекта (проектной группы в соответствии с 1.15);

-fo – ключ формализованного вывода (отсутствия форматирования при выводе) результата.

Результат выдается из базы данных «МСЦ-Кростат» и выводится в узло-часах. (сколько узло-часов потребил текущий пользователь в заданной системе за заданный период). Актуальность результата зависит от усановленной администратором СУППЗ периодичности обновления базы данных. Если указать опцию -fo, то команда mstat не будет выводить дополнительную информацию и форматировать вывод.

При указании единичной опции `-u` команда `mstat` выдаст статистику текущего пользователя за указанный период по всем проектам, например:

```
mstat –u –s clk 2020-03-01 2020-12-01
```

При указании опции `-p` без опции `-u` команда `mstat` выдаст статистику по заданному проекту с идентификатором `project_id` за указанный период. Выполняющий команду пользователь должен быть участником проекта (членом проектной группы в соответствии с 1.15). Пример команды для этого случая:

```
mstat –p my_project2 –s clk 2020-03-01 2020-12-01
```

При одновременном указании опций `-p` и `-u` команда `mstat` выдаст статистику по заданному проекту с идентификатором `project_id` за указанный период с выводом статистики по каждому пользователю, считавшему в рамках проекта. Выполняющий команду пользователь должен быть участником проекта (членом проектной группы в соответствии с 1.15). Пример команды для этого случая:

```
mstat –p my_project2 -u –s clk 2020-03-01 2020-12-01
```

3.4 Получение дополнительных возможностей СУППЗ

3.4.1 К дополнительным возможностям СУППЗ относятся:

- выделение ВМ решающего поля без запуска задания в соответствии с 3.4.2 необходимо в тех случаях, когда программисту требуется вычислительный ресурс в виде одного или несколько ВМ на определенное время, но при этом нет необходимости запускать конкретное задание или параллельную программу;

- работа с интерактивными заданиями в соответсвии с 3.4.3;

- поиск в СУППЗ задания пользователя по данным паспорта в соответствии с 3.4.4.

3.4.2 Для выделения ВМ решающего поля без запуска задания программист формирует к СУППЗ специальный запрос командой

```
getnodes [-s <имя_системы>] –np <число_ВМ>
         [-maxtime <время>] <имя_запроса>
```

где ключ `-s` задает имя системы в соответствии с 2.1.7;

`число_ВМ` – число запрашиваемых ВМ (не процессоров или ядер);

`время` – максимальное время, на которое запрашиваются ВМ;

`имя_запроса` – имя, которое будет присвоено запросу на требуемый вычислительный ресурс.

При выполнении команды `getnodes` запрос на требуемый ресурс будет поставлен в очередь, как обычное задание. При этом к имени запроса будут добавлены уникальный номер имя логической системы в соответствии с 2.1.8, т.е. сформирвано полное имя задания (запроса). Как и для обычного задания, в соответствии с 2.1.9 будет сформирован каталог ввода-вывода. После того, как запрос отстоит в очереди, СУППЗ выделит ВМ числом `число_ВМ` на указанное в параметре `-maxtime` время. Следует отметить, что правила применения и действие параметра `-maxtime` полностью соответствуют 3.1.1.1.

При успешном выполнении запроса в файл стандартного вывода задания (файл `output` в соответствии с 2.1.9) будет помещена информация о выделенных ВМ. С момента выделения ВМ, т.е. после прохождения запроса через очередь, любой из выделенных ВМ становится доступен программисту с помощью команд `ssh` или `rsh` (в зависимости от настроек, заданных системным программистом). Досрочное завершение работы с выделенными ВМ и снятие задания (запроса) с выполнения осуществляется в соответствии с 3.1.5.

3.4.3 При запуске задания в соответствии с 3.1.1.9 в команде `mpirun` может быть указан параметр `-interactive`, а при запуске задания в соответствии с 3.1.2, в секции `[Redirections]` файла-паспорта задания в соответствии с 4.1.1.5 может быть задан параметр `interactive`. Запущенное таким образом задание получает статус **интерактивного**, и для работы с ним необходимо соблюдать следующие правила.

3.4.3.1 После прохождения через очередь и запуска интерактивное задание при попытке чтения из стандартного потока ввода приостанавливается до тех пор, пока программист не подсоединится к заданию, выполнив команду

```
mattach [-s <имя_системы>] [имя_задания]
```

где `имя_задания` – полное имя задания в соответствии с 2.1.8, ключ `-s` задает имя системы в соответствии с 2.1.7

При отсутствии параметра в стандартный поток вывода будет выдан список всех выполняющихся заданий пользователя, и будет предложено выбрать задание для подсоединения. При успешном выполнении команда `mattach` связывает свои стандартные потоки ввода, вывода и ошибок с соответствующими стандартными потоками интерактивного задания. После подсоединения задание продолжит выполнение, при этом программист получает возможность в диалоге вводить требуемые заданию данные и просматривать его стандартный вывод.

3.4.3.2 Для интерактивного задания СУППЗ игнорирует заданные в соответствии с 3.1.1.6 – 3.1.1.8 или с 4.1.1.5 перенаправления стандартных потоков ввода, вывода и ошибок. Для перенаправления стандартных потоков своего интерактивного задания программист средствами операционной системы должен перенаправить стандартные потоки команды `mattach`.

3.4.3.3 Интерактивное задание считается выполняющимся с момента своего запуска, а не с момента выполнения команды `mattach`. Если программист не выполнит команду `mattach`, то по истечении заказанного времени задание будет снято. В этом случае, а также тогда, когда интерактивное задание само завершается до выполнения программистом команды `mattach`, весь произведенный процессами задания вывод в стандартные потоки вывода и ошибок будет записан в файлы `output` и `errors` в соответствии с 2.1.9.

3.4.3.4 Завершение команды `mattach` (например, по нажатию Ctrl-C) не приводит к завершению самого задания. Команда `mattach` может быть выполнена и завершена несколько раз, при этом работа с интерактивным

заданием каждый раз будет продолжаться с места, на котором была завершена предыдущая команда mattach. Вместе с этим одновременно для одного и того же задания программист может выполнить лишь одну команду mattach.

3.4.4 Поиск задания в СУППЗ по данным паспорта осуществляется командой

```
mfind <имя_секции> <имя_параметра> <маска_значения>
```

где имя_секции – имя секции паспорта задания в соответствии с 4.1.1;

имя_параметра – имя параметра указанной секции паспорта задания в соответствии с 4.1.1;

маска_значения – диапазон значений указанного параметра паспорта задания.

Команда mfind выдает в стандартный поток вывода полные имена заданий, паспорта которых соответствуют критериям поиска. Например, получить список всех заданий с именами, начинающимися на букву g, можно с помощью следующей команды

```
mfind General task_name 'g*'
```

Получить список всех заданий со временем выпонения 100 минут можно с помощью следующей команды

```
mfind TimeRequest limit 100
```

4 Входные и выходные данные

4.1 Входные данные

4.1.1 Паспорт задания СУППЗ

4.1.1.1 Поступающее в СУППЗ задание должно иметь оформленный паспорт, сохраненный во входном файле. Имя файла с паспортом задания должно быть передано в качестве параметра команде запуска задания в соответствии с 3.1.2. Для заданий, запускаемых в соответствии с 3.1.1, 3.1.2 паспорт будет сформирован автоматически, и никаких действий программиста по его офрмлению в этом случае не требуется.

4.1.1.2 Паспорт задания представляет собой текстовый файл, разделенный на секции. Название секции заключается в квадратные скобки и должно начинаться с первой позиции новой строки. Порядок следования секций в файле не имеет значения, за исключением, когда в файле присутствуют секции с одинаковыми именами. В этом случае к исполнению будет принята секция, указанная первой.

Как правило, содержимое секции составляют пары вида

```
параметр=значение
```

Если в значении строкового параметра присутсвуют пробелы, то такое значение должно быть заключено в двойные кавычки. Комментарии в паспорте допустимы, должны начинаться на #, и будут игнорироваться при разборе паспорта. Пример оформления секции паспорта задания:

```
# Это комментарий
# Следующая строка – начало секции с именем Section1
[Section1]
# Следующая строка – параметр par1 со значением 100
par1 = 100
# Следующая строка – параметр par2 со значением Hello, World!
# Поскольку значение содержит пробел, оно заключается в кавычки
par1 = "Hello, World!"
```

4.1.1.3 Обязательная секция `[General]` содержит следующие общие параметры задания:

- обязательный параметр `task_name` задает имя задания, которое служит основой для формирования полного имени задания в соответствии с 2.1.8;

- обязательный параметр `host_directory` определяет имя домашнего каталога задания, именно в этом каталоге при постановке в очередь в соответствии с 2.1.9 будет создан каталог ввода-вывода задания;

- обязательный параметр `cpu_count` определяет требуемое число процессоров (ядер) для выполнения задания;

- необязательный параметр `system` определяет имя логической системы в соответствии с 2.1.7, если параметр `system` не будет указан, задание будет направлено в логическую систему по умолчанию.

Пример оформления секции `[General]`:

```
[General]
# имя логической системы
system = mpi
# имя задания – testmod
task_name = testmod
# имя каталога задания
host_directory = /home/user/testmod/work
# число процессоров для выполнения задачи
cpu_count = 32
```

4.1.1.4 Обязательная секция `[TimeRequest]` содержит параметры задания, определяющие требования к времени выполнения:

- обязательный параметр `limit` определяет максимальное время выполнения задания (в минутах), после истечения этого времени задание будет принудительно завершено; сообщение о принудительном завершении задания будет выведено в журнал процесса-менеджера задания в соответствии с 4.2.6; указанное в параметре `limit` время будет участвовать в расчете приоритета задания в соответствии с 2.2.4.6 – 2.2.4.8;

- обязательный параметр `quant` определяет размер кванта (в минутах) для фоновых заданий в соответствии с 2.2.2 для пакетных и отладочных задач

значение параметра `quant` должно быть 0 или равно значению параметра `limit`;

- необязательный параметр `term_time` задает дополнительное время (в минутах) для завершения задания в соответствии с 2.2.6;

- необязательный параметр `term_signal` задает сигнал, который будет разослан всем процессам задания в качестве предупреждения о предстоящем завершении в соответствии с 2.2.6.2. Формат задания сигнала должен соответствовать команде ОС Linux `kill`.

Пример оформления секции `[TimeRequest]`:

```
[TimeRequest]
# максимальное время выполнения задания
limit = 200
# квант для фонового задания
quant = 20
# дополнительное время
term_time = 5
# сигнал о завершении
term_signal = USR1
```

4.1.1.5 Необязательная секция `[Redirections]` содержит параметры перенаправления стандартных потоков ввода, вывода и ошибок процессов задания:

- необязательный параметр `stdout` определяет имя файла, в который будет перенаправлен стандартный поток вывода процессов задания, если параметр не будет указан, перенаправление будет произведено в соответствии с 2.1.9;

- необязательный параметр `stderr` определяет имя файла, в который будет перенаправлен стандартный поток ошибок процессов задания, если параметр не будет указан, перенаправление будет произведено в соответствии с 2.1.9;

- необязательный параметр `stdin` определяет имя файла, на который

будет перенаправлен поток стандартного ввода процессов задания. На указанный файл в каталоге ввода-вывода задания автоматически будет создана символическая ссылка с именем `input`. Если параметр не будет указан, стандартный ввод для процессов задания будет недоступен и не должен использоваться в параллельной программе;

- необязательный параметр `interactive` может принимать значение `yes` или `no`. Указание значения `yes` делает задание интерактивным в соответствии с 3.4.3 и отменяет значения остальных параметров секции `[Redirections]`.

Пример оформления секции `[Redirections]`:

```
[Redirections]
# имя файла стандартного потока вывода задания
stdout = /home/user/test.out
# имя файла стандартного потока ошибок задания
stderr = /home/user/test.err
# имя файла стандартного потока ввода задания
stdin = /home/user/test.in
# задание не интерактивное
interactive = no
```

4.1.1.6 Обязательная секция `[Batch]` содержит текст командного файла сценария выполнения задания. Сценарий выполнения задания разрабатывается программистом на языке любого доступного на ВМ командного интерпретатора. В момент запуска задания из содержимого секции `[Batch]` СУППЗ автоматически сформирует командный файл `runmvs.bat`, расположенный в соответствии с 2.1.9. Командный файл `runmvs.bat` будет выполнен на первом ВМ из списка выделенных для задания ВМ.

При вызове командного файла сценария выполнения задания СУППЗ передаст в него следующие параметры:

- первый параметр – число процессоров (ядер), выделенных для задания;

- второй параметр – имя текстового файла со списком выделенных для

задания ВМ; каждая строка файла описывает один выделенный ВМ и имеет следующий формат:

`node_name:cpu_count`

где `node_name` – имя выделенного ВМ;

`cpu_count` – число процессоров (ядер) на ВМ.

- третий параметр – имя файла, в который программист при желании может записать в текстовом виде код завершения задания;

- четвёртый параметр – имя файла-паспорта задания, из которого программист может считать настройки, заданные как системой, так и самим программистом.

Используя переданные СУППЗ параметры, программист может предусмотреть произвольный порядок развертывания и выполнения процессов задания на выделенных ВМ.

4.1.1.7 Прогаммист может записать в паспорт собственные настройки задания в виде конфигурационных секций в соответствии с 4.1.1.2. СУППЗ сохранит эти настройки, и они будут доступны программисту в командном файле сценария выполнения задания – имя файла-паспорта задания будет передано в сценарий четвёртым параметром в соответствии с 4.1.1.6.

4.1.2 Конфигурационный файл пользовательских настроек СУППЗ

4.1.2.1 Пользовательские настройки СУППЗ сохраняются в конфигурационном файле с именем `.crunmvs`, который должен находиться в домашнем каталоге пользователя. Настройки определяют поведение по умолчанию клиентской утилиты СУППЗ и, следовательно, команд СУППЗ, рассмотренных в 3. Формат конифигурационного файла пользовательских настроек СУППЗ аналогичен формату паспорта задания СУППЗ, рассмотренному в 4.1.1.2. Рассмотрим секции конфигурационного файла.

4.1.2.2 Обязательная секция `[General]` содержит следующие общие параметры пользовательских настроек:

- необязательный параметр `tmp_directory` определяет имя каталога, в котором команды СУППЗ будут создавать временные файлы. Если параметр не

задан, или его значение неправильное (например, нет такого каталога, или нет доступа к обозначенному каталогу), то временные файлы будут создаваться в каталоге /tmp. Параметр применяется для обеспечения конфиденциальности работы пользователя с СУППЗ;

- обязательный параметр host определяет имя управляющей ЭВМ, с которой будет соединяться клиентская утилита runmvs для передачи команд СУППЗ серверу запросов; параметр host не принимается к исполнению, если указан параметр -host в команде mpirun в соответствии с 3.1.1.15.

Пример оформления секции [General]:

```
[General]
# имя каталога для временных файлов
tmp_directory = ~/my_home_tmp
# имя управляющей ЭВМ
host = head
```

4.1.2.3 Обязательная секция [Mpirun] содержит значения по умолчанию для следующих параметров команды mpirun, рассмотренной в 3.1.1:

- необязательный параметр quantum определяет значение по умолчанию параметра -quantum в соответствии с 3.1.1.2; если значение параметра quantum равно 0, то задания по умолчанию не будут фоновыми;

- необязательный параметр maxtime определяет значение по умолчанию параметра -maxtime в соответствии с 3.1.1.1;

- обязательный параметр stdiodir определяет значение по умолчанию параметра -stdiodir в соответствии с 3.1.1.5; перед использованием этот параметр будет обработан командным интерпретатором; категорически рекомендуется указывать значение 'pwd', чтобы каталог ввода-вывода задания создавался в каталоге, в котором запускается команда mpirun;

- необязательный параметр restart принимает значения 0 или 1; если указано значение 1, то по умолчанию будет действовать параметр -restart в соответствии с 3.1.1.3, т.е. задание после выполнения по умолчанию будет вновь поставлено в очередь;

- необязательный параметр `term_time` определяет значение по умолчанию параметра `–termtime` в соответствии с 3.1.1.10, указание значения 0 означает отсутствие дополнительного времени для завершения заданий;

- необязательный параметр `term_signal` определяет значение по умолчанию параметра `–termsignal` в соответствии с 3.1.1.11, указание значения 0 означает отсутствие сигнала для извещения процессов задания о предстоящем завершении.

Пример оформления секции `[Mpirun]`:

```
[Mpirun]
# квант времени для фонового задания
# задан ноль – это означает, что задание не является фоновым
quantum = 0
# время по умолчанию выполнения задания (задается в минутах)
maxtime = 300
# имя каталога задания
# если указано 'pwd' – то создавать каталог
# ввода-вывода задания в текущем каталоге
stdiodir = 'pwd'
# необходимость рестарта задания по его окончании
# 0 – рестарт не производить
restart = 0
# дополнительное время (в минутах) на завершение задания
# 0 – нет дополнительного времени
term_time = 0
# сигнал, посылаемый процессам задания для уведомления
# о предстоящем завершении
# 0 – нет сигнала
term_signal = 0
```

4.1.3 Файл-шаблон для задания разного числа MPI-процессов на разных ВМ

4.1.3.1 По умолчанию число процессов задания, запускаемых на ВМ в соответствии с 3.1.1, равно числу процессоров (ядер) ВМ, т.е. по одному процессу на одно ядро. Программист может самостоятельно задавать, сколько процессов задания на каком ВМ должно быть запущено. Для этого программист должен предположить, что его задание будет запущено на нужном ему числе ВМ с предопределенными именами `node1`, `node2`, `node3`,..., `nodeN`, где `N` – число необходимых для задания ВМ. Распределение процессов по ВМ задается с помощью специального файла-шаблона следующего формата.

4.1.3.2 Каждая строка файла-шаблона содержит описание одного ВМ. Заметим, что заданию при старте будет выделено столько ВМ (не процессоров!), сколько их задано в файле-шаблоне. ВМ в файле-шаблоне именуются предопределенными именами `node1`, `node2`, `node3`, и т.д. При этом номера модулей должны идти подряд без пропусков, т.е. нельзя, например, задать `node4` и `node6`, а `node5` пропустить.

4.1.3.3 В строке файла-шаблона после имени ВМ через двоеточие указывается число процессов задания на этом модуле. Если двоеточие отсутствует, число процессов принимается равным 1. В файле-шаблоне могут содержаться комментарии. Комментарий начинается с символа # и продолжается до конца строки. В качестве символа начала комментария может использоваться точка с запятой. Пример файла-шаблона:

```
# Это комментарий
node1:2     # Комментарий м.б.в строке с информацией
node2       # Если не указано число процессов на ВМ,
            # запускается один процесс
node03:004 # Могут присутствовать ведущие нули
node6:2     # Порядок следования ВМ
node5:2     # не имеет значения
```

4.1.3.4 Рассмотрим примеры неправильных строк.

```
module4:2  # Ошибка!!! Неправильное имя ВМ
node5:     # Ошибка!!! После двоеточия должно
# быть указано число процессов
```

4.1.3.5 После того, как файл-шаблон сформирован, его имя можно указать в качестве значения параметра `–machinefile` в команде `mpirun` в соответствии с 3.1.1.12. Заметим, что после запуска задания вместо ВМ с именами `node1–nodeN` заданию будут выделены реальные ВМ решающего поля с реальными именами.

4.1.3.6 Рассмотрим пример № 1 использования файла-шаблона. Создадим файл-шаблон с именем `example` и содержимым

```
node1:3
node2:2
```

При запуске программы `cpi` с помощью команды

```
mpirun –np 5 –machinefile example cpi
```

программа `cpi` запустится на двух ВМ, при этом на первом ВМ будут запущены три процесса, а на втором – два.

4.1.3.7 Рассмотрим пример № 2 неправильного использования файла-шаблона. Создадим файл-шаблон с именем `example` и содержимым

```
node1:3
node2:2
```

При запуске программы `cpi` с помощью команды

```
mpirun –np 6 –machinefile example cpi
```

программа `cpi` не запустится, поскольку в файле-шаблоне `example` указано 5 процессов, а в команде `mpirun` специфицировано большее значение – 6.

4.1.3.8 Рассмотрим пример № 3 использования файла-шаблона. Создадим файл-шаблон с именем `example` и содержимым

```
node1:3
node02
node03:10
```

При запуске программы `cpi` с помощью команды

```
mpirun –np 6 –machinefile example cpi
```

программа cpi запустится на трех ВМ, при этом на первом ВМ будут запущены три процесса, а на втором один, на третьем – два, всего – 6 процессов.

4.1.3.9 Рассмотрим пример № 4 использования файла-шаблона. Создадим файл-шаблон с именем example и содержимым

```
node1:3
node02:4
node03:4
```

При запуске программы cpi с помощью команды

```
mpirun –np 6 –machinefile example cpi
```

программа cpi запустится на двух ВМ (информация о третьем модуле не будет использоваться), при этом на первом и втором ВМ будут запущены по три процесса, всего – 6 процессов.

4.1.3.10 Рассмотрим пример № 5 использования файла-шаблона. Создадим файл-шаблон с именем example и содержимым

```
node1:2
node02
node01:1
```

При запуске программы cpi с помощью команды

```
mpirun –np 5 –machinefile example cpi
```

программа cpi запустится на двух ВМ, при этом на первом ВМ будут запущены четыре процесса с рангами 0, 1, 3 и 4, а на втором – один с рангом 2.

4.1.3.11 Рассмотрим пример № 6 неправильного использования файла-шаблона. Создадим файл-шаблон с именем example и содержимым

```
node1:2
node03
node04:2
```

При запуске программы cpi с помощью команды

```
mpirun –np 5 –machinefile example cpi
```

программа cpi не запустится, т.к. пропущено имя модуля node2.

4.1.3.12 Рассмотрим пример № 7 неправильного использования файла-шаблона. Создадим файл-шаблон с именем example и содержимым

```
node1:2
node03:3
node02:2
```

При запуске программы `cpi` с помощью команды

```
mpirun -np 5 -machinefile example cpi
```

программа `cpi` не запустится, т.к. для распределения процессов по ВМ будут использоваться только первые две строчки и `node02` окажется пропущенным.

4.1.4 Спецификация дополнительных ресурсов для задания

4.1.4.1 В соответствии с 1.6 в СУППЗ могут быть специфицированы дополнительные ресурсы для ВМ решающего поля. Имена и значения дополнительных ресурсов определяются в соответствии с 3.3.6. Программист может специфицировать необходимые для задания дополнительные ресурсы в виде текстового файла, который подается на вход команды `mpirun` в соответствии с 3.1.1.14. Файл-спецификация дополнительных ресурсов разрабатывается программистом в тех случаях, когда он хочет задать дополнительные ресурсы только для отдельных (не для всех) ВМ своего задания. Для спецификации ресурсов для всех ВМ удобнее воспользоваться параметром `-resource` команды `mpirun` в соответствии с 3.1.1.13.

4.1.4.2 Для задания спецификации дополнительных ресурсов программист должен предположить, что его задание будет запущено на нужном ему числе ВМ с предопределенными именами `node1`, `node2`, `node3`,..., `nodeN`, где `N` – число необходимых для задания ВМ. Ресурсная конфигурация модулей задачи задается с помощью специального файла-спецификации следующего формата.

4.1.4.3 Общий формат файла-спецификации аналогичен общему формату паспорта задания СУППЗ, рассмотренному в 4.1.1.2. Файл-спецификация состоит из нескольких секций, минимум из двух. Обязательная секция `[Resources]` содержит список вычислительных модулей с предопределенными именами `node1-nodeN`. Каждая строка секции `[Resources]` содержит предопределенное имя модуля и ссылку на ресурсную

секцию с описанием требований к ресурсам. Формат строки секции [Resources] следующий

```
имя_модуля = [имя_ресурсной_секции]
```

Если имя ресурсной секции не задано, это будет означать, что для данного модуля не заданы ресурсные требования. Ресурсные секции могут иметь произвольные имена, задаваемые программистом. Каждая ресурсная секция описывает дополнительные ресурсы, требуемые для того или иного ВМ. Заметим, что если для нескольких ВМ ресурсные требования совпадают, то для этих модулей в секции [Resources] можно указать одну и ту же ресурсную секцию. Рассмотрим пример файла-спецификации с комментариями.

```
[Resources]
# требования для модулей node1 и node2
# будут заданы в секции R1
node1 = [R1]
node2 = [R1]
# требования для модуля node3
# будут заданы в секции R2
node3 = [R2]
# Ресурсная секция R1
# требования к ресурсам для node1 и node2
[R1]
# Числовой ресурс
mem = 16
# Если значение строкового ресурса содержит
# пробелы, то оно заключается в кавычки
OS = "Linux RedHat"
# Ресурсная секция R2
# требования к ресурсам для node3
[R2]
# Числовой ресурс
```

```
mem = 32
# Строковый ресурс
OS = Windows
```

4.1.4.4 Файл-спецификация подключается к заданию либо с помощью параметра `-resourcefile` команды `mpirun` в соответствии с 3.1.1.14, либо путем прямого включения спецификации дополнительных ресурсов в паспорт задания при запуске с помощью команды `mrunf` в соответствии с 3.1.2.

4.1.4.5 При составлении и подключении файла ресурсной спецификации следует помнить, что размещение процессов задания на ВМ будет зависеть от параметра `-np` команды `mpirun`. Если число процессоров в ключе `-np` будет превышать суммарное число процессоров на ВМ из списка секции `[Resources]`, то система автоматически и по своему усмотрению дополнит список секции `[Resources]` до необходимого числа процессоров. Если число процессоров в ключе `-np` будет меньше суммарного числа процессоров на модулях из списка секции `[Resources]`, то система автоматически отбросит из списка секции `[Resources]` «лишние» модули.

4.1.4.6 Рассмотрим пример № 1 файла спецификации дополнительных ресурсов. Пусть конфигурация дополнительных ресурсов соответствует примеру из 3.3.6, и необходимо запустить задание на трех ВМ, первый из которых должен иметь не менее 16 Гб оперативной памяти. Создадим файл-шаблон с именем `myres` и содержимым

```
[Resources]
# Для первого модуля задаем
# ресурсные требования через секцию R1
node1 = [R1]
# Для node2 и node3 требования не заданы
node2
node3
# Ресурсная секция R1
# требования к ресурсам для node1
[R1]
```

```
# Числовой ресурс
mem = 16
```

В соответствии с 3.1.1.14 запускаем задание командой

```
mpirun –np 24 –resourcefile myres myprog
```

4.1.4.7 Рассмотрим пример № 2 файла спецификации дополнительных ресурсов. Пусть конфигурация дополнительных ресурсов соответствует примеру из 3.3.6, и необходимо запустить задание на трех ВМ, последний из которых должен иметь не менее 16 Гб оперативной памяти. Создадим файл-шаблон с именем `myres` и содержимым

```
[Resources]
# Для node1 и node2 требования не заданы
node1
node2
# Для последнего модуля задаем
# ресурсные требования через секцию R1
node3 = [R1]
# Ресурсная секция R1
# требования к ресурсам для node3
[R1]
# Числовой ресурс
mem = 16
```

В соответствии с 3.1.1.14 запускаем задание командой

```
mpirun –np 24 –resourcefile myres myprog
```

4.1.4.8 Рассмотрим пример № 3 файла спецификации дополнительных ресурсов. Пусть конфигурация дополнительных ресурсов соответствует примеру из 3.3.6, и необходимо запустить задание на трех ВМ, первый и последний из которых должны иметь не менее 16 Гб. Создадим файл-шаблон с именем `myres` и содержимым

```
[Resources]
# Для первого модуля задаем
# ресурсные требования через секцию R1
node1 = [R1]
```

```
# Для node2 требования не заданы
node2
# Для последнего модуля задаем
# ресурсные требования через секцию R1
node3 = [R1]
# Ресурсная секция R1
# требования к ресурсам для node1 и node3
[R1]
# Числовой ресурс
mem = 16
```

В соответствии с 3.1.1.14 запускаем задание командой

```
mpirun –np 24 –resourcefile myres myprog
```

4.1.4.9 Рассмотрим пример № 4 неправильного файла-спецификации. Модуль с одним и тем же именем дважды указан в списке секции `[Resources]:`

```
[Resources]
node1 = [R1]
node2
node1 = [R1]
[R1]
mem = 16
```

При запуске задания будет выдано сообщение об ошибке:

```
runmvsd: form_qimage: multiple entries of node node1 in section [Resources] of task image
```

4.1.4.10 Рассмотрим пример № 5 неправильного файла-спецификации. Задана несуществующая ресурсная секция:

```
[Resources]
node1 = [R3]
node2
[R1]
mem = 16
```

При запуске задания будет выдано сообщение об ошибке:

`runmvsd: form_qimage: missing section [R3] for node node1 in section [Resources]`

4.2 Выходные данные

4.2.1 Выходные данные СУППЗ для программиста представлены в следующих файлах каталога ввода-вывода в соответствии с 2.1.9:

- `errors` – файл, куда перенаправляется стандартный поток ошибок процессов задания;

- `.hosts` – файл со списком ВМ, выделенных для задания;

- `manager.log` – файл журнала менеджера задания СУППЗ.

4.2.2 В файл стандартного потока ошибок задания `errors` могут быть выведены диагностические сообщения:

- от коммуникационных и программных библиотек, используемых в параллельной программе из состава задания;

- от ОС ВМ об ошибочных и аварийных ситуациях, возникших при выполнении процессов задания;

- сообщения, предусмотренные самим программистом.

Если характер диагностических сообщений от программных библиотек или ОС ВМ не понятен программисту, ему следует обрабтиться за консультацией к оператору или системному программисту СУППЗ.

4.2.3 Файл со списком ВМ, выделенных для задания, является текстовым файлом со списком выделенных для задания ВМ. Каждая строка файла описывает один выделенный ВМ и имеет следующий формат:

`node_name:cpu_count`

где `node_name` – имя выделенного ВМ;

`cpu_count` – число процессоров (ядер) на ВМ.

4.2.4 Файл журнала процесса-менеджера задания не имеет формально заданной структуры. Тем не менее, в нем можно выделить следующие основные части:

- информация о подготовке задания к выполнению;

- информация о выполнении и завершении процессов задания;

- информация об освобождении ВМ после заверешения задания.

4.2.5 Рассмотрим пример файла журнала процесса-менеджера задания, приведенный на рисунке 3.

```
***********
Task started in 1 time of 1 at 10.11.14 09:50:13
Preparing nodes for user ant
****************** node1187 *********************
Host node1187 has been prepared
****************** node1186 *********************
Host node1186 has been prepared
Starting task process with pid 29933
Sleeper 29934 started at 10.11.14 09:50:14 for 5 minutes
Pid of sleeper is 29934
Signals set
SIGUSR2 received, killing sleeper
Killing sleeper 29934
manager: sleeper 29934 was completed on a SIGHUP, signal 12 has detected
Killing child tasks of pid 29933
manager: process 29933 done at 10.11.14 09:51:15
Creating done-file /usr/runmvs/users/ant/done/cpi.1
Done-file created successfully
10.11.14 09:51:15: manager: closing nodes
Close nodes started
Executing /usr/runmvs/bin/trycmd 3 /usr/runmvs/bin/cmd 540 /usr/bin/ssh on
node node1186
Executing /usr/runmvs/bin/trycmd 3 /usr/runmvs/bin/cmd 540 /usr/bin/ssh on
node node1187
Rsh on node node1187 Ok
Rsh on node node1186 Ok
close_nodes: node node1186 has been successfully freed
close_nodes: node node1187 has been successfully freed
close_node: close_nodes script done
10.11.14 09:51:21: manager: close_nodes done
Unlocking nodes ...
closing SLA quota...
Exiting...
```

Рисунок 3 – Пример файла журнала менеджера задания

Информация о подготовке задания к выполнению начинается со строки

```
Task started in 1 time of 1 at 10.11.14 09:50:13
```

которая обозначает начало запуска задания 10 ноября 2014 года в 9 часов 50 минут. Далее идет сообщение о начале подготовки ВМ к выполнению задания пользоввтеля `ant`:

```
Preparing nodes for user ant
```

Следующие строки сообщают о нормальном завершении процесса подготовки выделенных для задания ВМ с именами `node1186` и `node1187`. Информация о выполнении задания начинается со строки

```
Starting task process with pid 29933
```

и заканчивается сообщениями процесса-менеджера о нормальном завершении задания

```
manager: process 29933 done at 10.11.14 09:51:15
```

и об успешной регистрации факта завершения задания

```
Done-file created successfully
```

Со строки

```
Close nodes started
```

начинается информация о протекании процессов освобождения ВМ после заверешения задания. В примере из рисунка 3 ВМ с именами `node1186` и `node1187` успешно освобождены от процессов задания и возвращены в решающее поле со статусом «свободен». Последней строкой идет сообщение процесса-менеджера о своем собственном завершении.

4.2.6 Отдельное диагностическое сообщение предусмотрено для случая, когда задание исчерпывает заказанное время и принудительно снимается с выполнения. Этот случай проиллюстрирован на рисунке 4. Сообщение об исчерпании времени выглядит следующим образом:

```
**** WARNING!!! The task has exhausted settled time!!! ****
```

Если программист не обнаружил ожидаемых результатов выполнения задания в файле стандартного вывода или выходных файлах задания, рекомендуется проверить наличие указанного сообщения в файле журнала менеджера задания. Если подобное сообщение обнаружено, следует перезапустить задание с указанием большего времени выполнения.

```
***********
Task started in 1 time of 1 at 10.11.14 09:50:58
****************** node1188 **********************
Host node1188 has been prepared
Preparing nodes for user ant
/usr/runmvs/bin/prepare_nodes '/usr/runmvs/users/ant/queue/cpi.2'
Starting task process with pid 30305
Sleeper 30306 started at 10.11.14 09:50:59 for 1 minutes
Pid of sleeper is 30306
Signals set
Sleeper 30306 got up at 10.11.14 09:51:59, testing end of task
10.11.14 09:51:59: sleeper: msgget to qserver ok, sending message
10.11.14 09:51:59: sleeper: printflog done
10.11.14 09:51:59: sleeper: msgget from qserver ok, recieving reply

**** WARNING!!! The task has exhausted settled time!!! ****

manager: sleeper 30306 was comleted with code 4
Killing child tasks of pid 30305
manager: process 30305 done at 10.11.14 09:52:04
Creating done-file /usr/runmvs/users/ant/done/cpi.2
Done-file created successfully
10.11.14 09:52:04: manager: closing nodes
Close nodes started
Executing /usr/runmvs/bin/trycmd 3 /usr/runmvs/bin/cmd 540 /usr/bin/ssh on
node node1188
Rsh on node node1188 Ok
close_nodes: node node1188 has been successfully freed
10.11.14 09:52:09: manager: close_nodes done
Unlocking nodes ...
closing SLA quota...
Exiting...
```

Рисунок 4 – Пример файла журнала менеджера задания для случая, когда задание было принудительно снято с выполнения по причине исчерпани заказанного времени

5 Сообщения СУППЗ

5.1 Диагностические сообщения СУППЗ можно разделить на следующие типы:

- сообщения об ошибках при обращении к СУППЗ;
- системные сообщения об ошибках процессов обработки задания, которые могут быть выведены в:

а) файл стандартного потока ошибок задания;

б) файл журнала менеджера задания СУППЗ.

5.2 В случае получения системных сообщений действия программиста полностью определяются содержанием системной диагностики.

5.3 Таблица 1 содержит основные сообщения об ошибках и описание действий программиста при возникновении аварийных ситуаций при вызове команд СУППЗ.

Т а б л и ц а 1 – Диагностические сообщения при выполнении команд СУППЗ

Диагностика функции	Описание ситуации	Действия программиста
qserver for system default does not exist	Не инициализирован сервер очередей СУППЗ	Необходимо обратиться к оператору для запуска сервера очередей
the ordered number of processors does not correspond to the list of nodes	Неправильно задана ресурсная конфигурация задания	Необходимо проверить правильность задания ресурсной спецификации в соответствии с 4.1.4
invalid (missing) section name for node in section [Resources]	Для одного или нескольких ВМ, указанных в секции [Resources] файла-спецификации дополнительных ресурсов СУППЗ, отсутствует секция с описанием ресурсов	

Окончание таблицы 1

Диагностика функции	Описание ситуации	Действия программиста
requested resources for job do not exist	Не найдено необходимые ресурсы для для задания	Необходимо проверить правильность задания ресурсной спецификации в соответствии с 4.1.4
job start denied from directory	Для пользователя запрещен старт задания из каталога, не связанного с группой проекта в соответствии с 1.16	Необходимо запускать задания из каталога, связанного с группой проекта в соответствии с 1.15
access denied for user	Пользователь не внесен в список пользователей, которым разрешен запуск заданий	Необходимо обратиться к оператору для получения разрешения на запуск задания
client error on connect	Ошибка соединения с сервером запросов	Необходимо проверить правильность указания сетевого имени управляющей ЭВМ в пользовательском конфигурационном файле СУППЗ в соответствии с 4.1.2.2
error on prepare nodes	Ошибка подготовки ВМ к обработке задания	Необходимо обратиться к оператору или системному программисту
error on close nodes	Ошибка очистки ВМ после обработки задания	

Перечень терминов

В настоящем документе применены следующие термины с соответствующими определениями:

параллельная программа: программа, состоящая из нескольких взаимодействующих процессов, которые могут одновременно выполняться на разных процессорах;

задание (параллельное задание): информационный объект, в состав которого входят:

- параллельная программа, решающая прикладную задачу;
- паспорт задания;
- входные данные параллельной программы.

паспорт задания: информационный объект, представленный в виде файла и содержащий описание параллельного задания и требования к вычислительным ресурсам.

Перечень сокращений

СУППЗ – система управления прохождением параллельных заданий;

ВМ – вычислительный модуль;

ГПУ – графическое процессорное устройство

ОС – операционная система