ПРОГРАММНЫЕ МЕТОДЫ КОНТРОЛЯ И ДИАГНОСТИКИ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ В СИСТЕМАХ УПРАВЛЕНИЯ РЕСУРСАМИ

К. Е. Крамаренко

Сибирский государственный университет телекоммуникаций и информатики

УДК 519.676

В работе приведены результаты анализа вычислительной сложности контрольно-диагностических процедур в системе управления ресурсами SLURM, с целью определения пределов их масштабируемости – зависимости времени работы от количества вычислительных узлов в ВС. Выработаны рекомендации по оптимизации контрольно-диагностических процедур в пакете SLURM.

Ключевые слова: контроль, диагностика, система управления ресурсами, вычислительные системы.

Введение. Функционирование вычислительных систем (ВС) осуществляется под управлением систем управления ресурсами. Если проанализировать программный стек вычислительных систем из списка Top500 (редакция июнь 2017 года) [1], то можно заметить, что одном из самых распространённых пакетов управления ресурсов является SLURM [2]. Неотъемлемыми компонентами инструментария организации функционирования ВС являются подсистемы контроля и диагностики. Подсистема контроля предназначена для определения факта возникновения отказа ресурса ВС, а подсистема диагностики для локализации отказавших компонентов.

В современных ВС процедуры контроля и диагностики логически реализованы на различных уровнях:

– на уровне системы управления ресурсами, контролируется состояние узлов ВС;

– на уровне системы параллельного программирования, контролируется работоспособность процессов параллельной программы.

В данной работе анализируются программные методы контроля и диагностики на уровне системы управления ресурсами SLURM.

Центральным компонентом пакета SLURM является процесс *slurmctld*, который через фиксированные интервалы времени выполняет контрольно-диагностические тесты:

1. Определение сетевой доступности вычислительных узлов (*ping*).

2. Запуск диагностических тестов на вычислительных узлах.

Проверка сетевой доступности P вычислительных узлов осуществляется путем их последовательного опроса. Диагностические тесты представляют собой специализированные сценарии, подготавливаемые администратором системы для конкретного вычислительного узла, которые возвращают структуру описывающую его состояние.

В работе выполнен асимптотический анализ вычислительной сложности перечисленных контрольно-диагностических процедур с целью определения пределов их масштабируемо-

Работа выполнена при поддержке Российского фонда фундаментальных исследований (коды проектов 16-07-00712-а, 16-07-00992-а, 15-07-00048-а).

сти – зависимости времени работы от количества вычислительных узлов в ВС. Выработаны рекомендации по оптимизации контрольно-диагностических процедур в пакете SLURM.

1. Архитектура системы управления ресурсами SLURM. Как показано на рис. 1, система управления ресурсами (СУР) SLURM состоит из нескольких компонент. Главным компонентом системы является управляющий процесс *slurmctld*, который контролирует функциональное ядро системы. Другим компонентом системы является набор пользовательских команд, предоставляющих авторизованным пользователям возможность назначать задачи на узлы вычислительной системы и контролировать процесс их решение. Процесс вычислительного узла (ВУ) *slurmd* запускается на каждом вычислительном узле системы и в его функции входит решать задачи, наблюдать за ходом решения, собирать информацию о ресурсах вычислительного узла и передавать ее управляющему процессу.

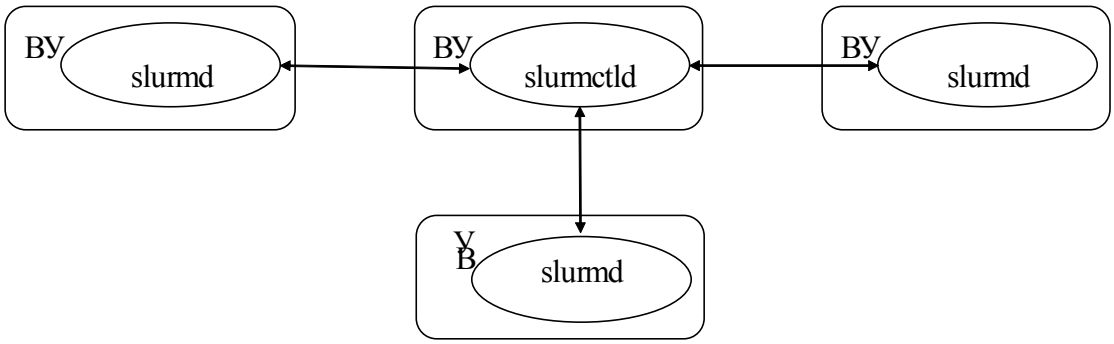

Рис. 1. Архитектура системы SLURM

2. Алгоритмы контроля и диагностики состояния узлов в системе SLURM. Основная работа, связанная с контролем и диагностикой состояния узлов вычислительной системы в SLURM, реализуется в отдельном потоке управляющего процесса *slurmctld*, выполняющем функцию SLURMCTLDBACKGROUND. Основными задачами этой функции являются: сохранение состояния управляющего процесса, диспетчеризация задач, контроль и диагностика состояния узлов вычислительной системы. Опишем алгоритмы, реализующие контроль и диагностику состояния узлов.

```
function SlurmctldBackground()
   while TRUE do
      if (TIME() – LastPingTime >= PingInterval) then
         LastPingTime = TIME()
         PINGNODES()
      end if
      if (TIME()– LastHealthCheckTime >= HealthCheckInterval) then
         LastHealthCheckTime = TIME()
         RunHealthCheck()
      end if
   end while
end function
```

Вычислительная сложность функции SLURMCTLDBACKGROUND равна

$$T_{SlurmctldBackground} = O\left(T_{PingNodes} + T_{RunHealthCheck}\right).$$

В главном цикле функции SLURMCTLDBACKGROUND каждые *PingInterval* секунд вызывается функция PINGNODES, задача которой состоит в проверке доступности узлов ВС по сети и регистрации узлов ВС, состояние которых СУР на данный момент неизвестно. Для этого формируется два списка запросов, первый содержит все узлы ВС доступность которых необходимо проверить, а второй содержит все узлы ВС состояние которых изменилось и в данный момент СУР не располагает актуальной информацией о них. После формирования списков, запросы добавляются в список запросов системы для дальнейшего их обслуживания. При получении запроса *Ping* вычислительный узел отвечает состоянием загрузки процессоров системы и объемом использованной оперативной памяти.

```
function PINGNODES()
    PingNodeList = LISTCREATE()
    RegNodeList = LISTCREATE()
    for each Node in NodeList do
        if (ISNODENOTREGISTERED(Node)) then
            LISTPUSH(RegNodeList, Node)
            continue
        end if
        if (ISNODENORESPOND(Node) OR ISNODEDOWN(Node)) then
            continue
        end if
        LISTPUSH(PingNodeList, Node)
    end for
    SENDREQUEST(PingNodeList)
    SENDREQUEST(RegNodeList)
end function
```

Вычислительная сложность функции PINGNODES равна

$$T_{PingNodes} = O\left(n + T_{SendRequest}\right),$$

где n – количество узлов в ВС, $T_{\text{SendRequest}}$ – время рассылки служебной информации с применением древовидного алгоритма, имеет порядок O(log$n$).

Также в главном цикле функции SLURMCTLDBACKGROUND каждый *HealthCheckInterval* секунд вызывается функция RUNHEALTHCHECK в задачи которой входит более детальная проверка состояния узлов ВС, чем в функции PINGNODES. При получении такого запроса вычислительный узел выполняет скрипт проверки системы (пользователь может сам определить какой скрипт выполнять, указав его имя в конфигурационном файле *slurmd.conf*), данный скрипт может производить диагностику сети, файловой системы, определять аппаратные сбои и сообщать об этом управляющему процессу *slurmctld*.

```
function RunHealthCheck()
    HealthCheckNodeList = LISTCREATE()
    for each Node in NodeList do
        if (ISNODENORESPOND(Node) OR ISNODEDOWN(Node)) then
            continue
        end if
        LISTPUSH(HealthCheckNodeList, Node)
    end for
```

SENDREQUEST(HealthCheckNodeList)
end function

Вычислительная сложность функции RUNHEALTHCHECK равна

$$T_{RunHealthCheck} = O\left(n + T_{SendRequest}\right),$$

где n – количество узлов в ВС, $T_{\text{SendRequest}}$ – время рассылки служебной информации с применением древовидного алгоритма, имеет порядок O(log$n$).

Заключение. Выполненный анализ алгоритмов контроля и диагностики системы SLURM показал, что время их выполнения линейно зависит от количества ЭМ в системе. В большемасштабных ВС целесообразен переход к более эффективным алгоритмам, например, основанным на децентрализованной схеме или с распараллеливанием основных фаз алгоритмов контроля и диагностики.

Список литературы

1. Top500 supercomputing sites [Электрон. ресурс]. URL: https://www.top500.org/ lists/2016/11/ (дата обращения 15.05.2017).

2. SLURM workload manager [Электрон. ресурс]. URL: https://slurm.schedmd.com/ documentation.html (дата обращения 01.03.2017).

Крамаренко Константин Евгеньевич – ст. препод. Сибирского государственного университета телекоммуникаций и информатики;
630102, Новосибирск; e-mail: kostya.kram@gmail.com