# ОПЫТ ПОСТРОЕНИЯ НЕБОЛЬШОГО ВЫЧИСЛИТЕЛЬНОГО КЛАСТЕРА

Андрей Е. Шевель
Andrei.Chevel@pnpi.spb.ru,
Петербургский Институт Ядерной Физики, Россия

Jerome Lauret
jlauret@mail.chem.sunysb.edu, университет Стони Брука, штат НьюЙорк, США

## *Аннотация*

Эта статья посвящена процессу построения вычислительного кластера для группы ядерной химии (Nuclear Chemistry Group - NCG) в государственном университете Стони Брука (штат НьюЙорк, США) [http://www.sunysb.edu/]. NCG участвует в работе коллаборации ФЕНИКС (PHENIX) [http://www.phenix.bnl.gov/] в Брукхевенской Национальной Лаборатории (Brookheven National Laboratory - BNL [http://www.bnl.gov]) на ускорителе RHIC [http://www.rhic.bnl.gov/RCF]. Коллаборация PHENIX записывает на магнитные ленты около половины Петабайта (1PB=1024TB) экспериментальной информации в год (500 Терабайтов - TB). Нашей задачей была организация вычислительной установки для физического анализа экспериментальных данных из коллаборации PHENIX для небольшой группы физиков 3-7 человек. Физический анализ производимый данной группой является малой частью работы по анализу данных производимых всей коллаборацией.

Обсуждаемая вычислительная установка была введена в строй в 2000-ом году.

## *ВВЕДЕНИЕ*

NCG вовлечена в экспериментальные работы в RHIC/Phenix с 1998 года. Фактически, многие небольшие физические группы, которые связаны с анализом физических данных, пытаются использовать вычислительные ресурсы там, где они уже существуют. Например, в других университетах, исследовательских центрах и т.п. Группа NCG решила использовать другой путь - построить свой собственный небольшой вычислительный кластер используя имеющийся предшествующий опыт построения кластеров такого вида для экспериментов в области физики высоких энергий. Естественно, в первую очередь был принят во внимание опыт коллаборации PHENIX и вычислительного центра ускорителя RHIC [http://www.rhic.bnl.gov/RCF], а также опыт, полученный в работе [http://www.hep.net/chep98/PDF/43.pdf]. Предполагалось, что создаваемый вычислительный кластер должен быть не просто большим калькулятором, но достаточно интеллектуальным инструментом по обработке физических данных.

Поскольку группа NCG может выполнять лишь малую часть физического анализа, который производится коллаборацией, то это означает, в частности, что должно использоваться то же самое программное обеспечение для анализа и моделирования, которое используется коллаборацией. Естественно, что упомянутое программное обеспечение может быть использовано на тех аппаратных и операционных платформах, которые поддерживаются коллаборацией. Таким образом, выбор был определён естественным стремлением сделать трудозатраты на построение вычислителя и его дальнейшую поддержку минимальными. Мы использовали архитектуру, которая стала фактическим стандартом для организации научных вычислений: использовали

центральную машину с традиционной операционной системой и системными приложениями и ряда периферийных вычислителей на основе двухпроцессорной материнской карты L440GX+ (Intel) под Linux.

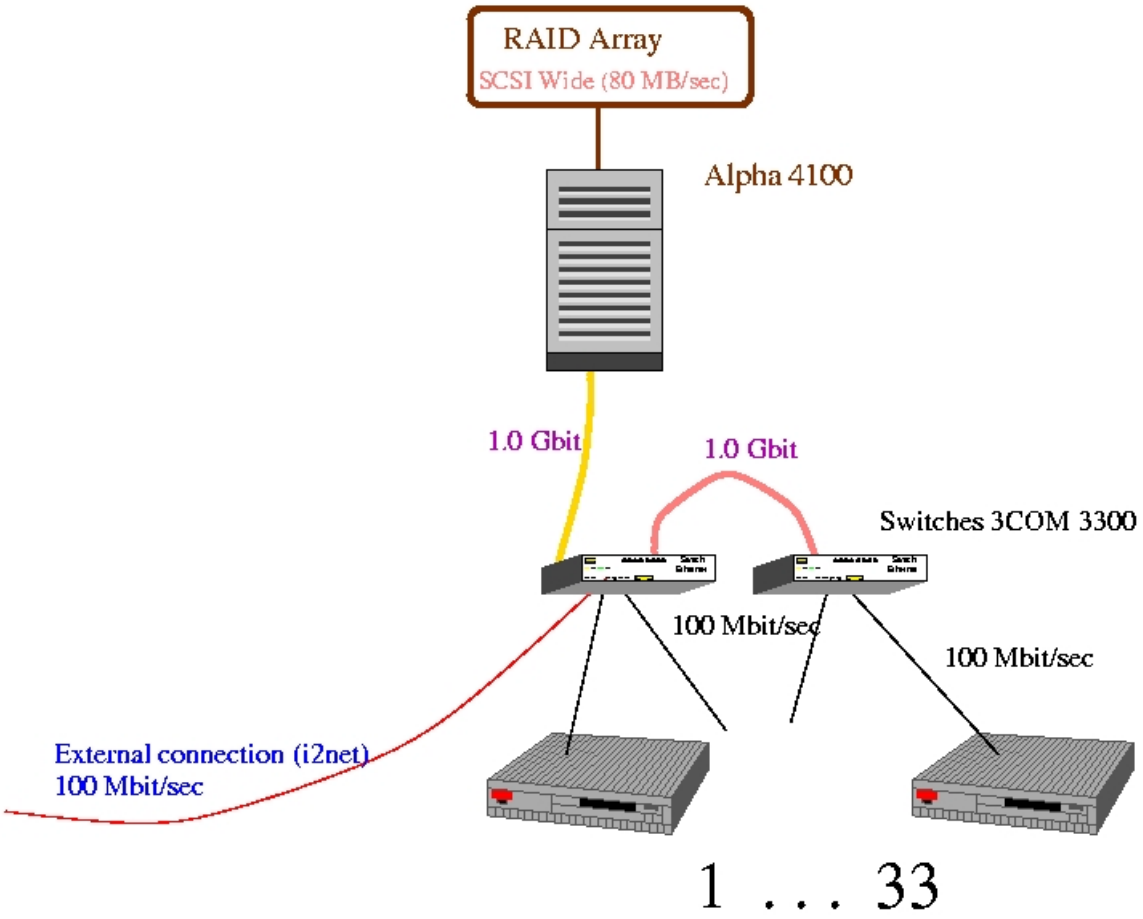


**Рисунок 1. Общая схема обсуждаемого кластера.**

Архитектура кластера приведена на рисунке 1. Основная идея такой архитектуры весьма прозрачна: использование каждого компонента в его наиболее естественной форме. Центральный компьютер Alpha 4100 используется в основном для традиционных, хорошо известных задач и функций: файл сервер для относительно большого объёма данных, выполнение регулярного резервного копирования критических данных, обеспечение безопасности вычислительной установки от атак из Интернет, поддержка X-сессий и т.д. С другой стороны, периферийные компьютеры под Linux являются главной вычислительной мощностью. Известно, что такого вида кластеры поставляются крупными компьютерными поставщиками, такими как IBM, HP и другими. Нашей задачей было найти приемлемый баланс между объёмом ручной работы по созданию кластера и дальнейшей эксплуатацией, а также объёмом доступных финансовых ресурсов.

В следующих разделах мы постараемся изложить наиболее важные с нашей точки зрения моменты разработки обсуждаемого вычислительного ресурса.

# *ОБОРУДОВАНИЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ*

## ОБЩИЕ СООБРАЖЕНИЯ

Кластерная схема вычислителя, составленная из недорогих покупных вычислительных элементов, позволяет наращивать вычислительную мощность по мере появления финансовых возможностей. Здесь можно отметить, что машины на базе процессоров Intel доступны во всём мире (практически в любом компьютерном магазине). Такая доступность и массовость определяет относительно невысокие значения показателя стоимость/производительность. Поскольку компьютерная мощность процессоров CISC (имеются в виду прежде всего Intel) удваивается каждые 16-18 месяцев, то важным является возможность оперативной модернизации периферийных машин или их отдельных узлов. Наконец, в процессоре с архитектурой CISC готовые к исполнению программы меньше по объёму занимаемой памяти (имеется в виду исполняемый код) в 1.5-2.5 раза в сравнении с процессорами с архитектурой RISC, что также имеет значение для приложений в физике высоких энергий ввиду больших объёмов исполняемого кода (многие десятки MB программного кода).

Ускорение вычислений с использованием кластерной архитектуры при обработке данных экспериментов по физике высоких энергий получается практически в ***n*** раз, где ***n*** есть число периферийных машин. Такое происходит благодаря особенностям потока данных, который состоит из последовательных статистически независимых *событий* (групп чисел), каждая из которых (группа) описывает отдельное физическое событие.

## ВЫБОР ОБОРУДОВАНИЯ

Мы получили две стойки с периферийными вычислителями (по 16 узлов в каждой стойке с предустановленной системой RedHat Linux 6.1), которые были поставлены компанией VALINUX [http://www.valinux.com/].

В качестве центрального узла мы нуждались в машине, которая могла бы удовлетворять ряду требований.

- Приемлемую производительность: производительность процессора и пропускная способность каналов ввода/вывода.
- Масштабируемость: сервер должен иметь возможность дальнейших расширений во всех направлениях.
- Со стороны компании поставщика должна быть обеспечена соответствующая поддержка и модернизация как по оборудованию, так и по программному обеспечению.

Нетрудно видеть, что таким требованиям удовлетворяют машины из разных компаний: Sun, HP, IBM и других. В нашем случае был приобретён сервер Alpha 4100. Основные характеристики сервера приведены в [1,2]. Среди них полезно отметить следующие.

- Пропускная способность шины компьютера составляет около 1.1 GByte/sec при средней скорости доступа к основной памяти 750 Mbyte/sec.
- Машина имеет 16 PCI слотов; каждый из 4-х каналов PCI (64 битная шина) способен передавать данные со скоростью 250 MBytes/sec.
- Сервер имеет один процессор Alpha EV6 533 MHz CPU (можно расширить до 4-х процессоров).
- Основная память имеет объём 2 GB (можно расширять).
- Два дисковода (DEC RZ2DD-LS; 9GB и IBM DDYS-T18350M; 18GB) установлены в специальном раке BA36R-SD UltraSCSI StorageWorks с возможностью горячей замены.

- 4 интерфейса Ethernet, один 10/100 Mbit и три оптических канала Ethernet со скоростью 1 Gbit.
  - Замечание. Канал 100 Mbit в настоящее время не используется. Один оптический канал используется для связи с периферийными машинами и для выхода во внешнюю сеть Интернет-2. Два других оптических канала находятся в процессе конфигурирования с целью увеличения пропускной способности сети между центральной машиной и периферийными вычислителями.
- 2 канала SCSI 80 Mbytes/sec (в настоящее время используется один для связи с массивом RAID ёмкостью 3 TB).

Каждый периферийный вычислитель включает следующее.

- Два процессора Pentium III (Katmai), 500MHz.
- Материнская карта L440GX+.
- 512 MB основной памяти.
- Два порта Ethernet по 100 Mbit.
- SCSI disk drive = QUANTUM Model: ATLAS IV 9 WLS, 9 GB, 80MB/sec.
- SCSI adapter = Adaptec AIC-7896/7 Ultra2 SCSI host adapter.

Кроме перечисленного оборудования были приобретены необходимые периферийные устройства.

- Ленточная библиотека ***TLS-4660-4LV2***: поставлена компанией Qualstar [http://www.qualstar.com/] основанной на технологии SONY AIT [http://www.aittape.com/]. Устройство ***TLS-4660-4LV2*** содержит 60 слотов (мест для картриджей), имеет 4 магнитофона (возможно расширение до 6 магнитофонов). Каждый картридж может содержать 50 GB неуплотнённых данных. Каждый магнитофон может читать или писать со скоростью 6 MB/sec. Таким образом, вся ленточная библиотека позволяет записать 3 TB данных с общей скоростью 24 MB/sec.
- Дисковый массив RAID: мы приобрели устройство ***Cobra C-2 Enterprise*** в компании **RaidInc** [http://www.raidinc.com/], массив из дисководов 32x36 GB + 16x72 GB. В сумме полезная ёмкость составляет 2.2 TB. Массив имеет два контроллера (ведущий/ведомый). Общая теоретически достижимая пропускная способность составляет 80 MB/sec.

## ВЫЧИСЛИТЕЛЬНАЯ АРХИТЕКТУРА

Как легко понять из предыдущего изложения, наша вычислительная система является централизованной. Ниже продолжим обсуждение важных особенностей периферийных машин, работающих под Linux.

Каждая периферийная машина имеет свою собственную копию операционной системы включая все системные библиотеки и дополнительное свободно распространяемое программное обеспечение. Каждый периферийный узел имеет одинаковую (или весьма близкую) конфигурацию, включая каталоги ***/tmp*** и ***/scratch***. В то же время, все пользовательские файлы, компоненты прикладного программного обеспечения хранятся в дисковом массиве RAID, т.е. монтируются на каждой периферийной машине посредством протокола **NFS3**.

Все узлы в нашем кластере являются частью domain **NIS**. Такое решение было принято, чтобы обеспечить полную прозрачность для пользователя; пользователь "видит" свои данные и программы вне зависимости от машины, на которой он (пользователь) в данный момент использует систему (логирован). Известно, что **NIS** весьма активно использует протокол **RCP**. Так, во время старта задания на периферийном компьютере, **NIS** порождает довольно мощный поток запросов **RCP**. Если задания стартуют сразу на многих машинах, то такой поток может привести к медленному старту заданий. Следовательно, **NIS** может уменьшить производительность системы на коротких заданиях. Однако, в нашем случае мы имеем поток заданий со

средним временем выполнения (от 10 минут до 1.5 часа). Эта потенциальная проблема отмечалась также в [3].

Кластер подключён к backbone ***Internet-2*** [http://www.internet2.edu/]. Эта новая сеть является сетью с ограниченным доступом и она менее опасна в отношении сетевых атак, чем обычный Интернет. Тем не менее, в последующих разделах обсуждается ряд мер по предотвращению несанкционированного доступа к компьютерной системе.

## СИСТЕМНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

Сервер Alpha 4100 был поставлен с операционной системой *TruUnix64 4.0f* (ранее имела название *Digital Unix*). Сервер начал работать в конце 1999. На периферийных машинах под Linux *Redhat 6.1* было установлено то же ядро, что используется в **BNL**.

В качестве распределённой файловой системы была установлена *afs* [http://www.alw.nih.gov/WWW/AFS-resources.html]. Необходимость определялась тем, что *afs* серверы в **BNL** хранят всё программное обеспечение для коллаборации PHENIX. В дополнение, хранилище CVS также располагается в пространстве *afs*. Использование *afs* позволяет поддерживать локальную копию всего необходимого программного обеспечения. Локальная копия обновляется автоматически на регулярной основе. Что касается реализации *afs*, то мы использовали версию ***Transarc AFS***, которая была предоставлена **BNL**. В то же время имеется другая версия *afs* ***OpenAFS*** [http://www.openafs.org/]. Мы не исключаем, что в будущем может потребоваться переход на ***OpenAFS***.

Мы также установили ряд программных компонентов из проекта **GNU** в обеих системах (Linux и Tru64). Было обращено особое внимание на то, чтобы на обеих платформах были установлены одинаковые версии таких ключевых пакетов как **gcc**, **grep**, **sed**, **make**. Полезно обратить внимание, что поскольку обсуждаемая вычислительная установка есть часть крупномасштабной работы, то авторы имеют ограниченную свободу в выборе версий базовых продуктов.

После обзора базовых компонентов нашего кластера можно перейти к детальному рассмотрению отдельных проблем, которые потребовалось решить.

## *СОХРАННОСТЬ ДАННЫХ*

Для поддержки любых вычислительных установок имеются как минимум две основные задачи: безопасность (сохранность данных любых видов, включая программы) и стабильность работы. Одна сторона безопасности есть поддержание целостности данных. Другая сторона – предотвращение несанкционированного доступа к вычислительным системам. Речь идёт не только о вторжении по сети, но и о физическом вторжении, в том числе непреднамеренном. Автор был свидетелем случая, когда уборка соседнего помещения привела к заливанию вычислительного кластера водой и только случайность предотвратила полное затопление и выход из строя дорогостоящего оборудования.

Ниже мы обсудим схемы поддержания целостности данных, которые были использованы в настоящем проекте.

## ТЕХНОЛОГИЧЕСКОЕ ОКРУЖЕНИЕ ВЫЧИСЛИТЕЛЬНОГО КЛАСТЕРА

Термин ***безопасность*** используется здесь в широком понимании. Здесь имеется в виду целый ряд мер начиная от правильно организованной вентиляции и кондиционирования воздуха, отдельного помещения с весьма ограниченным доступом для размещения оборудования вычислительного кластера и необходимой периферии.

Естественно, что файл сервер и массив дисковой памяти подключены к устройству бесперебойного питания. Мы также проследили, чтобы на нашем электрическом фидере не было каких-то других мощных потребителей электроэнергии.

Перечисленные в этом разделе аспекты общей проблемы безопасности вычислений рассматривались нами как критически необходимые.

## РЕЗЕРВНОЕ КОПИРОВАНИЕ

В предыдущем разделе была упомянута ленточная библиотека ***TLS-4660-4LV2***. Эта библиотека используется вместе с программной системой ***Networker***, которая была поставлена компанией ***Legato*** [http://www.legato.com/], в схеме резервного копирования. Что касается содержания резервных копий, то было решено копировать только часть наших файлов. Копируются следующие компоненты: все пользовательские каталоги, критические системные и конфигурационные файлы. Не копируются области, где располагаются физические данные из-за их большого объёма. Инкрементальное копирование производится каждый день. Раз в неделю – полная копия. Мы храним примерно полугодовой цикл полных копий.

Естественно, что резервное копирование производится в автоматическом режиме и, как правило, вмешательство человека не требуется. Такая схема резервного копирования даёт основательные гарантии, что файлы не пропадут бесследно из-за выхода из строя дисковода или (что более вероятно) из-за ошибки менеджера. Типичная полная копия имеет объём более 100GB, который распределён по более, чем 1200K файлов.

Среди прочих предпринятых нами мер по увеличению стабильности работы файл-сервера полезно упомянуть файловую систему **AdvFS** в операционной системе **True64**. Лишь упомянем несколько полезных свойств, которые заметно упрощают задачи администрирования во вторичной памяти.

- Использование механизма *мусорной корзины* при удалении файлов.
- Широкие возможности по реконфигурированию логических томов, в том числе *во время использования этих томов*.
- Встроенные средства резервного копирования.

В нашем случае эти дополнительные средства резервного копирования используются для копирования индекса базы данных основной системы резервного копирования **Networker**. Дело в том, что выход из строя дисковода может стать причиной повреждения индекса базы данных системы копирования, что приведёт к весьма длительной (много часов) процедуре восстановления индекса. Использование дополнительной схемы резервного копирования позволяет избежать потерь времени даже при пропаже критических конфигурационных файлов самой системы **Networker**. Вторичная схема резервного копирования реализована в виде задания системы **cron**, которое выполняется каждые двое суток.

## УМЕНЬШЕНИЕ РИСКА НЕСАНКЦИОНИРОВАННОГО ДОСТУПА

Следующий аспект безопасности состоит в предотвращении несанкционированного доступа к вычислительному ресурсу по компьютерной сети. В нашем случае это было тем более важно, поскольку кластер располагается в университете, который довольно часто оказывается мишенью атак различного рода по сети.

В качестве первого шага мы выключили все старые и небезопасные демоны в системе (**ftpd**, **telnetd**, **fingerd** и т.д.) и потребовали, чтобы все пользователи входили в систему с использованием клиентов на основе протоколов **ssh** (**sftp**, **scp**, **ssh**). Кроме того, вход на любую периферийную машину извне можно только с центральной машины (Alpha 4100). На каждой машине под Linux были задействованы все механизмы по ограничению числа IP адресов, откуда возможен вход на данную машину (файлы */etc/hosts.deny*, */etc/hosts.allow*, */etc/ssh*).

На центральной машине дважды в месяц автоматически просматриваются записи о всех входах в систему (файлы auth.log и daemon.log), готовится и рассылается отчёт. В отчёте содержится сортировка записей о входах в систему по нескольким параметрам: числу входов отдельного пользователя, мест, откуда производились входы, другая сопутствующая информация, включая сортировку всех неуспешных входах. Хотя, по опыту авторов, записи в упомянутых логах не всегда исчерпывающи или даже противоречивы, тем не менее такой отчёт даёт неплохое представление о попытках несанкционированного доступа. В дополнение, для анализа вторжений используется утилита **logcheck** [http://www.psionic.com/abacus/logcheck].

Следующая линия защиты от сетевых вторжений базируется на детектировании любых изменений системных конфигурационных файлов и исполняемых компонентов (скрипты и исполняемые программные коды). Хотя мы используем такие средства сканирования как **cops** [http://www.fish.com/cops/], **tiger** [http://net.tamu.edu/network/tools/tiger.html], **SATAN** [http://www.cs.ruu.nl/cert-uu/satan.html], было также применено разработанное нами средство сканирования всех критических файлов на предмет изменения контрольной суммы, прав доступа, дата создания/модификации прочие параметры каждого файла из наблюдаемой группы каталогов. Легко догадаться, что в наблюдаемую группу каталогов входят /etc, /bin, /usr/bin, /sbin, /usr/sbin и т.д. Естественно, что любой каталог сканируется рекурсивно, включая все подкаталоги. Любые изменения докладываются менеджеру. Такая процедура оказалась исключительно полезной просто для сопровождения конфигурационных файлов и системных программ, т.к. даёт менеджеру информацию о последних изменениях, что немаловажно учитывая ограниченность персонала. Особенно удобно это средство в нашем случае, когда лица ответственные за разные части системы находятся на значительном удалении один от другого и от самого вычислительного кластера (на сотни и тысячи километров). Программный пакет с аналогичными функциями известен как **Tripwire** [http://www.tripwire.com/] и поставляется как часть **Red Hat Linux**, однако, мы полагаем, что наша разработка более точно учитывает конкретные условия.

Посредством заданий **cron** несколько скриптов периодически сообщают о текущем состоянии вычислительной системы. Неполный список периодически производимых проверок содержит следующее.

- Поиск файлов, которые никому не принадлежат или имеют sticky uid.
- Проверка паролей пользователей системы на тривиальность.

Перечисленный комплекс мер рассматривается как необходимый минимум для обеспечения приемлемого уровня безопасности в отношении вторжений из компьютерной сети. Можно предпринять и другие шаги для повышения безопасности. Таким образом, безопасность – это никогда не заканчивающаяся история.

## ЛОКАЛЬНАЯ КОПИЯ ПРИКЛАДНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Обсуждаемый кластер предназначен в основном для обработки физических данных класса mDST в эксперименте RHIC/PHENIX. Естественно, что мы должны использовать для обработки данных точно то же программное обеспечение, что используется другими физиками в **BNL**. Как уже упоминалось, доступ к прикладному программному обеспечению в **BNL** легко реализуется посредством распределённой файловой системы *afs*. Однако, для удалённых потребителей такой способ использования программ и библиотек программ оказался довольно медленным и нестабильным. Некоторые из причин, которые приводили к перерывам в работе *afs*, были очевидны для нас: перерывы из-за проблем с сетевым периметром безопасности в **BNL**, окольная сетевая маршрутизация, прочее. Во многих других случаях было трудно составить точную картину того, что происходит в сети, почему возникает снижение пропускной способности на протяжении нескольких минут или часов.

Суммарный эффект от всех имеющихся задержек был таков, что сборка программы с использованием удалённой библиотеки могла занять несколько часов. Если же мы имели локальную копию библиотеки, то сборка занимала несколько минут. Имея перед собой такую картину, мы решили реализовать полуавтоматическую зеркальную копию всего прикладного программного обеспечения из эксперимента **PHENIX**. Естественно, были отобраны только две платформы для копирования: True64 и Linux. Общий объём реплики имеет порядок величины около 30 GB.

Обратим внимание, что столь незначительный объём всего программного обеспечения **PHENIX** для выполнения физического анализа может быть легко размещён на современном нотебуке.

Здесь хотелось бы сделать небольшое отступление по поводу коннективности между удалёнными университетами и **BNL**. Реальная скорость передачи данных с использованием команды **scp** между Университетом и **BNL** колеблется от 80 до 200 KBytes/sec. С помощью специализированного средства многопоточной передачи данных **bbftp** [http://ccweb.in2p3.fr/bbftp/] автору удалось увеличить скорость передачи до 2.9 MBytes/sec при передаче группы файлов общим объёмом 40 MB. Близкие значения были получены совершенно независимо в университете Вандербильта и коллегами из Японии. Полезно обратить внимание, что такая скорость передачи имеет место, несмотря на пропускную способность опорной сети (заявлено около 100 Mbytes/sec). В конкретном случае авторы ожидали достичь как минимум 10 Mbytes/sec, учитывая локальную коннективность 100 Mbit. Эксперты в коммуникационных отделах в BNL и в Университете сообщили нам, что им эта проблема известна, однако удовлетворительного решения пока не найдено (август 2001).

Подход к копированию всего программного обеспечения **PHENIX** был довольно простым: определённые области дерева *afs* в **BNL** сканировались разработанным на *perl* скриптом. Для каждого файла из дерева *afs* выполнялись обычные проверки, которые требуются для процедуры зеркального копирования. Логические линки в дереве *afs* правильно распознаются и создаются в соответствующих местах локальной копии. Единственной проблемой было автоматическое преобразование скриптов, которые создают стандартный для **PHENIX** набор переменных окружения (несколько более ста переменных). К счастью оказалось, что скрипты легко анализируются и преобразуются в вид, который пользователи обсуждаемого кластера применяют при входе в систему.

В начале планировалось использовать процедуру зеркального копирования раз в неделю. Позже, после тестирования и обсуждения, выяснилось, что удобнее использовать такую процедуру тогда, когда возникает реальная необходимость. Поскольку прикладные программы менялись довольно быстро, часть пользователей кластера хотела бы дистанцироваться от неизбежных новых ошибок в постоянно меняющемся программном обеспечении и использовать только стабильные версии прикладных программ.

В обсуждаемом контексте важным элементом была реализация локальной копии базы данных Objectivity/DB [4]. Эта СУБД имеет свой встроенный механизм копирования базы данных. Однако, в нашем случае мы не могли его использовать из-за наличия периметра сетевой безопасности **BNL**. В результате нами была разработана другая система реплицирования базы данных Objectivity/DB. Основная идея для реализованного нами механизма была высказана Satoshi Yokkaichi (RIKEN/CCJ), в которую мы внесли несколько улучшений. На сегодняшний день схема копирования представляет собой следующее.

В качестве исходного материала для копирования используется основная копия базы данных **Phenix** Objectivity/DB, хранящейся в **BNL** (база данных **Phenix** представляет собой ряд файлов распределённых по нескольким серверам). В процессе создания основной копии, или мастер-копии, все файлы объединяются таким образом,

что созданная мастер-копия содержит все файлы базы данных независимо от их первоначального расположения. Эта мастер-копия переносится на обсуждаемый кластер посредством команды **scp**. Объём копии составляет около 60 MB. После выполнения копирования создаётся новая (локальная) федеративная база данных (разумеется, сама СУБД должна быть развёрнута до этого). Затем, восстанавливается мастер-копия во вновь созданную федерацию. Наконец, необходимо запустить два демона на хосте, где восстановлена база данных: *oolock* и *ooams*. В своей основе все перечисленные действия с базой данных основаны на использовании следующих команд СУБД Objectivity/DB: *oobackup*, *oonewfd*, *oorestore*, *ooinstallfd*, *oolockserver*, *oostartams* и т.д. [4]. Как легко догадаться, все перечисленные действия описаны в скрипте, который в настоящее время запускается вручную по мере необходимости.

Поскольку мы имеем сублицензию из **BNL** на версию Objectivity/DB под Linux, то СУБД развёрнута на одной из периферийных машин.

## *СИСТЕМА ПАКЕТНОЙ ОБРАБОТКИ*

История систем пакетной обработки заданий (**batch systems**) насчитывает 40 лет или около того. Было разработано и исследовано масса алгоритмов, как делить доступные компьютерные ресурсы между многими заданиями. Было написано море статей с описанием математических моделей с использованием аппарата математической теории массового обслуживания. Наиболее популярные алгоритмы распределения ресурсов реализованы во множестве коммерческих и свободно распространяемых системах пакетной обработки для вычислительных кластеров.

Поскольку обсуждаемый кластер не слишком сложен и предназначен для относительно небольшого числа пользователей, то наши критерии распределения заданий по периферийным машинам были весьма просты: одно задание на процессор, что в нашем случае означает два задания на периферийную машину. Такой простой критерий, как оказалось, не создаёт проблем ни для менеджеров, ни для пользователей. В качестве системы пакетной обработки заданий была использована свободно распространяемая (вместе с исходными текстами) система Portable Batch System (**PBS**) [http://www.openpbs.org/]. Эта система была разработана национальным агентством по аэронавтике **NASA** много лет назад. Система **PBS** доступна в настоящее время как минимум в двух вариантах: **Open PBS** (свободно распространяемая) и **Professional PBS** (коммерческий продукт). В нашем кластере использован свободно распространяемый вариант. Такое решение потребовало разработки нескольких скриптов, чтобы гарантировать, что никакой пользователь не сможет монополизировать кластер для своих задач в ущерб остальным пользователям. Пришлось ввести ограничение на запуск заданий, если загрузка центрального сервера является слишком большой (в качестве такового мы рассматриваем ситуацию, когда команда uptime даёт величину загрузки равной 15.0 или более).

Все остальные возможности открытого варианта PBS оказались не идеальными, но приемлемыми для небольшого кластера с максимальным потоком в 1-2 тысячи заданий в день.

Хотелось бы перечислить несколько других систем пакетной обработки заданий, на которые мы обратили внимание при выборе подходящей.

- Network Queueing System (**NQS**) [http://www.gnqs.org/]: свободно распространяемая система.
- Load Sharing Facility (**LSF**) [http://www.lsf.com]: коммерческий продукт; хорошо продуманная система с большим количеством различных полезных свойств. Не дешёвая.
- **FBSNG** [http://www-hppc.fnal.gov/fbsng/] - Next Generation of Fermi Batch System; относительно новая свободно распространяемая система (по меньшей мере для некоммерческих применений); у авторов возникли сомнения по поводу возможности поддержки такой системы без помощи разработчиков длительное

время. Система написана на языке Python, однако к системе прилагается библиотека программ на C, которая зависит от версий интерпретатора Python. Не совсем ясно, кто будет модернизировать эту библиотеку вслед за версиями интерпретатора.

- Sun Gridware [http://www.sun.com/gridware]: свободно можно скачать готовые к исполнению программы. Неплохая система, она была приобретена компанией Sun около года назад. Ранее система поставлялась как коммерческий продукт под названием CODINE. Авторам не совсем ясно, как долго компания планирует поставлять эту систему бесплатно.
- Информацию о многих других пакетных системах, включая CONDOR, можно найти на странице [http://www.cmpharm.ucsf.edu/~srp/batch/systems.html].

Ясно, что настройка компонентов программного обеспечения для вычислительного кластера (никогда не заканчивающаяся работа) требует нетривиальных усилий. Посему, как только какая-то часть настройки завершена, то исключительно полезно записать все технические детали настройки в виде скрипта, который можно будет просто выполнить в будущем, когда это станет необходимым. В противном случае, при встрече подобных или просто тех же самых проблем придётся всё начать с нуля. В принципе, почти все несколько раз повторяющиеся действия должны быть реализованы в виде скриптов, даже если эти скрипты будут состоять из нескольких строк. Естественной целью таких действий является экономия рабочего времени и живого труда. Мы должны ясно понимать, что менеджеры могут приходить и уходить, но опыт настройки кластера должен накапливаться. Иными словами, следует строить стабильную работу в нестабильной обстановке. Чтобы реализовать это, необходимы соответствующие организационные механизмы (воплощённые в скриптах и заданиях **cron**).

## СОПРОВОЖДЕНИЕ КЛАСТЕРА И ОРГАНИЗАЦИОННЫЕ ВОПРОСЫ

В настоящее время (2001) достигнута относительно стабильная работа вычислительного кластера (перевызов системы производится примерно раз в месяц), который представляет собой отличный инструмент для физического анализа и который может быть использован небольшой группой физиков.

Есть ли проблемы с поддержкой работоспособности кластера? К сожалению, такие проблемы имеют место.

Во-первых, это общие проблемы, которые имеют место на любой реальной вычислительной установке: выходят из строя дисководы, материнские карты, источники питания и т.д.

Другой класс проблем, который требует внимания, - это смена версий используемых программных пакетов. Нет никакой необходимости устанавливать новые версии так же часто, как они появляются. Однако как минимум раз в год для каждого пакета вам необходимо установить новую версию, чтобы поддержать имеющиеся возможности на должном уровне, или предоставить пользователям новые возможности, которые требуются на новом этапе анализа данных.

Наравне с общими имеются более специфические проблемы, которые определяются конкретными параметрами компьютеров и природой прикладных программ. В частности, мы встретили ситуацию, когда во время выполнения потока заданий на кластере центральный сервер имел мало загруженный процессор (50% или около того) и, одновременно, высокую загрузку показывала команда uptime (25.0 и выше). В это время любая команда выполнялась столь медленно, что делало невозможным использование центрального сервера для продуктивной работы.

## ПОИСК УЗКИХ МЕСТ

Взглянем снова на рисунок 1. Кластер имеет 33 периферийных машины, которые могут генерировать поток данных около 100 Mbit/sec*33, т.е. суммарный поток может достигать величины 3.3 Gbit/sec. Эта величина более чем в три раза превышает пропкусную способность канала между сетевым коммутатором 3COM 3300 и сервером Alpha 4100 (1 Gbit/sec). Можно ожидать, что для ряда задач это может оказаться узким местом. Такова была первая идея, где искать узкое горло. Чтобы дать возможность работать остальным пользователям, средствами **PBS** было уменьшено допустимое число заданий в стадии выполнения (с 66 до 20). Загрузка центрального сервера снизилась до приемлемых величин (12.0 – 20.0).

Позже была предпринята попытка измерить сетевой трафик с помощью подручных средств. Было обнаружено, что типичное задание, которое состоит в вызове системы **ROOT** [5] и интерпретации скрипта с использованием **CINT** [5], открывает от 6 to 20 файлов в смонтированной посредством **NFS** файловой системе. Используемые файлы (**Linux**) легко видеть командой

**ls -l /proc/*n*/fd**

где ***n*** означает номер процесса для исследуемого задания. В то же время реальный сетевой трафик меняется во время выполнения задания и не превышает 3.3 Mbytes/sec на любой машине под Linux в соответствии со значениями, которые мы получили посредством пакета **ntop** [http://www.ntop.org], во время выполнения длинной последовательности заданий состоящей из 500 заданий. Другими словами, суммарный трафик не мог превышать 3.3 * 33 = 108.9 Mbytes/sec, следовательно, канал между центральным сервером и коммутатором, не может рассматриваться как перегруженный. Это верно, даже с учётом того факта, что **ntop** даёт не точные значения, а лишь оценки сетевого трафика, которые по нашему опыту могут отличаться от реальных значений в два раза.

Другим возможным узким местом может быть канал между дисковым массивом RAID и центральным сервером. Канал имеет максимальную пропускную способность 80 MB/sec (около 800 Mbit/sec). Далее, пришлось изучить особенности настройки операционной системы True64, которые описаны на страницах [6]. Здесь было найдено масса информации по настройке системы. После нескольких экспериментов и изучения вывода программы **sys_check** были увеличены значения параметра **name-cache-size** в ядре системы. Загрузка кластера сразу серьёзно уменьшилась, что позволило снова увеличить число заданий в стадии исполнения, увеличив тем самым производительность кластера.

Нелишне заметить, что любое изменение параметров ядра часто требует перезагрузки системы, которая не может быть произведена в любой момент на работающей машине, которую использует много людей находящихся в различных, часто весьма удалённых городах и странах. Таким образом, процесс настройки может занять несколько дней или даже несколько недель.

Примерно такое же рассуждение почти целиком приложимо к периферийным машинам под управлением **Linux.**

Повторим, что после затрат значительного времени на поиски и устранение возникающих проблем, очень полезно конвертировать полученный опыт в один или несколько скриптов, которые сразу показывают необходимые переменные или устанавливают их. Как правило, мы разрабатывали два вида скриптов.

- Скрипты для установки и развёртывания отдельных программных компонентов.
- Скрипты для слежения за ситуацией в системе (посредством заданий **cron**) или установки параметров.

Общий объём разработанных скриптов составляет около 10K строк.

## ДОБАВЛЕНИЕ НОВЫХ ПЕРИФЕРИЙНЫХ МАШИН

В процессе установки программного обеспечения на периферийных машинах мы постепенно наращивали число скриптов для установки программных компонентов. Когда число скриптов достигло 5, потребовалась некоторая стандартизация. Некоторые программные продукты устанавливались с использованием простейшей последовательности

```
configure
make
make install
```

Эта последовательность, которой предшествует автоматическая распаковка устанавливаемого пакета, была реализована в виде отдельного скрипта с параметрами. Однако, около половины установленных продуктов нуждались в дополнительных корректировках и не могли быть полностью настроены простейшей последовательностью. Обычно требовались такие дополнительные действия:

- изменения (настройка) параметров;
- копирование отдельных файлов в специальные места;
- финальная модификация, чтобы добиться соответствия с другими пакетами.

Естественным образом возникали скрипты для установки отдельных программных компонентов. Для интеграции всех инсталляционных скриптов использовался специальный каталог (**InstallScripts**) и мастер скрипт (**GenInstall**) для интерпретации содержимого упомянутого каталога. В каталоге могут быть любые файлы с любыми именами. Мастер-скрипт обращает внимание лишь на имена с форматом

**S***nnn_W_name*

Здесь

- *nnn* есть номер скрипта, который состоит точно из трёх цифр; номер определяет порядок интерпретации: первым интерпретируется скрипт с номером 000, последним – максимальный из имеющихся номеров; если номера одинаковы, то первым выполняется скрипт, который первым следует по алфавиту; нумерация для центрального сервера и для периферийной машины независимы друг от друга;
- *W* есть параметр, значением которого может быть **L** (выполнять в Linux) или **C** (выполнять на центральном кластере);
- *name* есть любое имя, которое вы хотите дать скрипту.

Любые имена файлов в каталоге, которые не удовлетворяют описанному формату, просто игнорируются скриптом **GenInstall**.

## БОРТОВОЙ ЖУРНАЛ

Ни один из авторов обсуждаемого проекта не может целиком посвятить своё время только рассматриваемому кластеру. Более того, авторы также распределены географически и не имеют возможности встречаться каждый раз, когда требуется что-то исправить или просто изменить. Сдедовательно огромную важность имеет возможность отслеживания изменений, который вносит тот или другой автор. Иными словами, требуется поддерживать компьютеризированный бортовой журнал, в которой авторы записывают вносимые изменения с описанием всех использованных команд и их параметров.

В данном проекте для бортового журнала использовалась специально разработанная доска объявлений, которую можно видеть посредством обычного браузера www. Мы записывали происходящие важные на наш взгляд события в системе

и вносимые изменения до такой степени деталировки, которая была достижима технически. Трудно переоценить пользу такого журнала, особенно когда вы встречаетесь с реальной проблемой. Объём журнала рос со временем одновременно с его ценностью. Естественно имеется возможность найти какие-то ключевые слова в данном журнале. В настоящее время объём журнала составляет более 6 MB.

## ПРОЦЕДУРА РЕГИСТРАЦИИ НОВЫХ ПОЛЬЗОВАТЕЛЕЙ

Среди организационных вопросов можно упомнить регистрацию новых пользователей, которая реализована в виде полуавтоматической процедуры. Потенциальный пользователь логируется на кластер с использованием протокола **ssh**, в качестве оболочки используется регистрационный скрипт. Пользователь отвечает на ряд вопросов. Среди прочего он должен ввести имя, телефон и мейловский адрес своего руководителя. Менеджеры автоматически уведомляются о любых попытках регистрации. Если пользователь ввёл всю необходимую информацию, то копия автоматически направляется его руководителю. Администрация в течение одного-двух рабочих дней уведомит пользователя о своём решении по поводу регистрации.

Заключая обсуждение административных проблем, можно суммировать, что как минимум два важных элемента должны присутствовать, если вам надо поддерживать такой кластер при минимуме рабочей силы:

- любой полученный полезный технический опыт с программами или оборудованием или любые изменения в программах и оборудовании должны быть описаны в специальном журнале (файле или группе файлов); журнал должен быть доступен всем лицам, кто несет ответственность за правильную работу вычислительной установки;
- любые действия, которые требуется выполнять более, чем один раз, следует записать в виде в скрипта.

# *ВОЗМОЖНЫЕ И РЕАЛИЗУЕМЫЕ НАПРАВЛЕНИЯ ДЛЯ БУДУЩИХ УЛУЧШЕНИЙ*

## ИНТЕГРИРОВАНИЕ ПРОГРАММНОЙ ИНФРАСТРУКТУРЫ

Учитывая масштабы вычислительного процесса в коллаборации PHENIX, мы рассматриваем обсуждаемый вычислительный кластер как отдельный элемент более крупной сети для физического анализа. Было найдено весьма полезным протестировать относительно новое программное обеспечение для интеграции разнообразных вычислительных ресурсов на более высоком уровне, чем обычное удалённый вход в систему (логирование), или простая пересылка файлов по сети. Имеется в виду система с архитектурой **grid**, в особенности система **Globus** [http://www.globus.org/]. В контексте Globus описываемый кластер рассматривается как один элемент (узел) вычислительной сети коллаборации.

Инструментальные средства системы **Globus** были развёрнуты на кластере и протестированы с аналогичной системой (**Globus**) в Петербургском Институте Ядерной Физики, а также с тестовой установкой **Atlas** в **BNL**. В настоящее время прототип вычислительной фабрики на основе описанного кластера функционирует нормально. В соответствии с нашим видением проблемы, **Globus** будет в ближайшие пару лет отличной базой для организации универсального решения более или менее стабильного набора потребностей для удалённого потребителя в коллаборации **PHENIX**.

- Сетевая безопасность.
- Передача данных.
- Удалённое управление файловой системой.

- Удалённый запуск заданий на любой системе пакетной обработки.
- Информационный сервис с использованием протокола **ldap**.

Особое значение это может иметь для лиц, которым требуется использовать сразу несколько вычислительных кластеров в разных концах города, страны или планеты с различными особенностями вычислений и различными данными. В обработке данных в физике высоких энергий нетрудно представить, что различные наборы данных полученных посредством моделирования или данные измерений (mDST) могут располагаться на различных кластерах. Для конечного пользователя проще послать задание туда, где находятся данные, а не наоборот. Естественно, что одно и то же задание, но с разными данными может выполняться на разных кластерах.

Поскольку сама система Globus не слишком критична по своим запросам к памяти или процессору, то её можно установить на большинство нотебуков.

## МОДЕРНИЗАЦИЯ ОБОРУДОВАНИЯ

Предшествующее обсуждение оборудования касалось существующего положения. В ближайшем будущем ситуация может меняться. Например, если заменить периферийные процессоры на более мощные, то сетевой трафик, предположительно, вырастет пропорционально росту мощности процессора. Замена процессоров – неплохая идея, поскольку в настоящее время использование процессора заданиями составляет 85%-95%. Кроме того, как мы полагаем, может измениться характер самих заданий.

Можно ожидать, что у нас будет от 30 GB до 100 GB на один набор данных (так называемый mDST) для физического анализа. На основании предыдущего опыта известно, что последовательность заданий для обработки различных порций одного набора mDST требует около 10 часов астрономического времени. Нетрудно видеть, что сама передача данных на периферийные машины занимает немного времени (100GB/80MB/sec), что примерно составит 20 минут. Можно заметить, что в наших условиях последовательность заданий выполняется на 2/3 периферийных машин кластера.

Таким образом, учитывая, что периферийные процессоры загружены на 90% или около того, большая часть времени выполнения заданий уходит на счёт, а не на передачу данных.

Имеется несколько способов модернизации аппаратной конфигурации.

- Увеличить число каналов SCSI для массива **RAID** (уже делается).
- Реализовать частную сеть, которая не имеет контактов с Интернет и локальной сетью университета, для трафика **NFS** (в процессе настройки).
- Заменить процессоры на более производительные (изучение описания материнской карты L440GX+ показало, что карта позволяет установить Pentium III с максимальной частотой 850 MHz, это в 1.7 раза быстрее имеющегося процессора).

## *ЗАКЛЮЧЕНИЕ*

Основная цель авторов состояла в создании вычислительной установки, которая могла бы функционировать практически автономно с минимумом ручной работы при приемлемой стоимости оборудования и программного обеспечения. Несколько механизмов (скриптов) дают объективную информацию менеджеру о состоянии и происходящих изменениях в системе, помогают развернуть новые программы и оборудование. Перечисленные меры дают возможность предотвратить большинство проблем и неплановых прерываний нормальной работы на небольшом вычислительном кластере.

Авторы полагают, что их цель в значительной степени достигнута.

Авторы уверены, что описанная реализация может быть использована как более или менее типовое решение для небольшой группы физиков, занимающихся анализом поступающих данных. Изложенный опыт может быть отправной точкой для проектирования новых вычислительных установок с подобными целями. После того как описываемый кластер уже начал использоваться, авторы встретили неплохое описание системы для управления кластером в [http://www.redbooks.ibm.com/redbooks/SG246041.html]. Кроме того, хорошим источником информации по современному пониманию проблем вычислительных кластеров являются труды конференции [http://www.ncsa.uiuc.edu/LinuxRevolution/], состоявшейся в июне 2001 года.

Авторы не смогли бы успешно завершить обсуждаемую разработку без обсуждений различных аспектов встреченных проблем с различными специалистами такими как: Behzad Barzideh, Bruce Gibbard, Terry Healy, Roy Lacey, Shigeki Misawa, Edward Nicolescu, Martin Purschke, Momchil Velkovsky, Satoshi Yokkaichi и многими другими.

## ССЫЛКИ

1. AlphaServer 4000/4100 Systems Technical Summary
   http://www.compaq.com/alphaserver/download/4100_summary.pdf
2. Alpha System 4100
   http://www.compaq.com/alphaserver/archive/4100/4100_tech.html
3. T. Sterling, J. Salmon, D. Becker, D. Savarese “How to Build a Beowulf” 1999, MIT, page 135.
4. Objectivity/DB Administration Release 5.2, 1999.
5. Пакет ROOT http://root.cern.ch
6. Improving the system performance (Представление URL слишком длинно, потому его пришлось разбить на две строки)
   http://www.tru64unix.compaq.com/faqs/publications/base_doc/DOCUMENTATION/V40F_HTML/AQ0R3GTE/MPPRFXXX.HTM