

# Анализ эффективности распределения задач суперкомпьютера\*

А.Л. Головинский<sup>1</sup>, А.А. Московский<sup>2</sup>, А.Л. Маленко<sup>1</sup>

Институт кибернетики им. В.М. Глушкова НАН Украины<sup>1</sup>, РСК Технологии<sup>2</sup>

При проектировании суперкомпьютера встает вопрос выбора оптимального алгоритма планирования очереди задач, который наиболее эффективно распределяет задачи между его узлами. Обычно планировщик выбирается разработчиком или администратором кластера на основании своего опыта, опыта пользователей или путем натуральных испытаний. В работе предложен формальный критерий сравнения планировщиков: индекс ожидания ресурсов, который служит оценкой качества предоставления вычислительных сервисов. Данный индекс применен для сравнения популярных планировщиков: Torque FIFO, Slurm Backfill и Slurm FIFO. Предложенный авторами подход к оценке качества кластера внедрен в систему управления суперкомпьютером SCMS 4.0.

## 1. Введение

При проектировании высокопроизводительной вычислительной системы возникает проблема выбора оптимального алгоритма планирования очереди вычислительных задач, который наиболее эффективно распределяет задачи, которые запускаются на данном кластере, между его узлами.

Обычно планировщик выбирается разработчиком или администратором кластера исходя из собственного опыта, опыта пользователей или путем натуральных испытаний разных планировщиков. Однако, чтобы получить статистически значимый результат натурального испытания, необходимо проводить его длительный период времени. На кластере, который используется реальными пользователями, отсутствует возможность сравнивать работу планировщиков на одном наборе задач, а поэтому сложно применять формальный критерий сравнения, поскольку результат зависит от конкретного набора задач. Кроме этого, результаты сравнения планировщиков, полученные в ходе натуральных испытаний на одном кластере, могут быть неприменимыми к другому.

Для решения этой задачи авторами был разработан стенд для моделирования поведения суперкомпьютера с кластерной архитектурой. На одном мощном вычислительном узле стенд позволяет моделировать до 1000 вычислительных узлов, а также значительно ускорять время выполнения виртуальных задач без существенной потери точности. Стенд создан на основе технологии виртуализации, которая использует незначительное количество процессорных ресурсов для работы каждого виртуального узла.

Входными данным для стенда является набор из  $n$  задач, которые описываются тройками чисел  $\{(q_i, r_i, c_i), i = 1, \dots, n\}$ , где  $q_i$  – время постановки задач в очередь,  $r_i$  – время выполнения,  $c_i$  – количество узлов, необходимых задаче для выполнения. Такие данные на реальном кластере получаются после анализа статистики выполнения задач, которую предоставляет планировщик кластера. В результате работы стенда получают моменты начала работы каждой задачи  $\{s_i, i = 1, \dots, n\}$ . Эти данные в дальнейшем используются для анализа качества работы планировщика.

Авторам удалось смоделировать поведение планировщика на кластере с пиковой производительностью до 380 ТФлопс, используя лишь один мощный узел (4 восьмиядерных процессора с частотой 3 ГГц), что эквивалентно производительности кластеров из 50-60 мест списка TOP-500 по состоянию на ноябрь 2011 года [1]. Для анализа результатов экс-

---

\*Выражаем благодарность коллегам из Южно-Уральского университета за предоставленные данные для экспериментов.

перимента создано специальное ПО.

Основные характеристики стенда:

- виртуализация до 1000 виртуальных узлов на одном реальном вычислительном узле;
- дополнительные реальные узлы позволяют масштабировать стенд до 10000-50000 виртуальных узлов;
- масштабирование времени в 20-50 раз;
- возможность установки разных планировщиков и их сравнение на одном наборе задач;
- автоматический сбор и обработка результатов запусков виртуальных задач.

## 2. Индексы оценивания качества сервиса

Качество предоставления сервиса высокопроизводительных вычислений можно определять разными путями. Это может быть наличие разнообразного прикладного ПО, удобство управления собственной очередью задач и файлами, скорость обмена данными между кластером и терминалом пользователя, надежность работы кластера в целом и т.д. В данной работе мы сосредоточимся лишь на одном аспекте качества кластера, который измеряется численно, – длительности ожидания ресурсов.

Кластер с неограниченным количеством узлов должен мгновенно обрабатывать запрос пользователя на получение ресурсов и сразу же запускать задачу на выполнение. Понятно, что в реальной ситуации такого не происходит. Реальный кластер имеет систему очередей задач и специальные алгоритмы для их управления. Каждый алгоритм по-своему определяет приоритетность запуска каждой задачи.

Часто у пользователей возникает необходимость контролировать ход выполнения задач, особенно в начале запуска, чтобы убедиться в отсутствии ошибок. В таких случаях время ожидания задачи в очереди, с точки зрения пользователя, потеряно бесцельно. В данной работе мы сформулировали индексы, которые количественно выражают среднее время ожидания ресурсов вычислительными задачами. Эти индексы можно вычислять и сравнивать для разных кластеров и планировщиков на одном наборе задач.

При рассмотрении индексов мы исходили из таких неформальных соображений. "Удобство" сервиса в нашем понимании состоит в некотором компромиссе между временем, которое приемлемо ожидать перед получением разрешения на выполнение задачи, и самим объемом полученных ресурсов (временем выполнения задачи и количеством узлов). Сложно указать точную формулу, сколько должен ждать пользователь, и какая граница времени ожидания является приемлемой. Следующие соображения являются попыткой получения этого компромисса.

Рассмотрим набор задач  $\{(q_i, s_i, r_i, c_i), i = 1, \dots, n\}$  за исследуемый период  $[0, T]$ , которые запускаются на  $N$ -узловом кластере. Считаем, что время выполнения задач больше одной временной единицы. Это логично, поскольку на практике основную массу ресурсов потребляют задачи с большим временем выполнения.

Введем следующие индексы. Среднее время ожидания:

$$W = \frac{1}{n} \sum_{i=1}^n (s_i - q_i),$$

среднее время ожидания, взвешенное на время выполнения задачи:

$$W_1 = \frac{1}{n} \sum_{i=1}^n \frac{s_i - q_i}{r_i},$$

среднее время ожидания, взвешенное на порядок длительности выполнения:

$$W_2 = \frac{1}{n} \sum_{i=1}^n \frac{s_i - q_i}{\ln r_i}.$$

Все индексы в той или иной мере описывают время ожидания ресурсов. Меньшее значения индекса означает лучший сервис.

Рассмотрим сначала индекс  $W$  – обычное среднее арифметическое времени ожидания. Недостатком этого индекса является то, что он не учитывает, что вероятность выделения ресурсов убывает при увеличении количества запрашиваемых узлов. С другой стороны, для ”длинных” задач, которые работают, например, неделю, ожидание один или два часа не влияет на комфортность восприятия сервиса.

Для более корректного сравнения мы вводим индексы  $W_1$  и  $W_2$ , которые показывают количество времени, которое ожидает задача, нормированное на время ее выполнения или порядок длительности выполнения.

Идея введения индексов  $W_1$  и  $W_2$  простая. Они лишены одного недостатка индекса  $W$ , а именно: учитывается время работы задачи. Но  $W_1$  и  $W_2$  не учитывают размер задачи. Ниже вводится индекс, который будет учитывать и этот фактор. Для его вычисления необходимо получить оценку вероятности освобождения необходимого количества ресурсов.

Обозначим  $\xi(t)$  – случайный процесс, который означает количество занятых ресурсов в момент  $t$ ,  $\xi(t) \in \{0, \dots, N\}$ , где  $N$  – общее количество узлов кластера. Процесс  $\xi(t)$  является кусочно-постоянным со случайными моментами прыжков. В момент запуска задачи процесс увеличивается на ее размер, в момент окончания – соответственно уменьшается.

Для упрощения будем считать, что распределение свободных ресурсов в каждый момент времени  $t$  не зависит от  $t$ . Обозначим  $F(x)$  – функцию распределения количества занятых узлов  $\xi(t)$ . Тогда вероятность немедленного запуска задачи на  $c$  узлах равняется вероятности, что количество занятых узлов не превышает  $N - c$ , что в свою очередь равно  $F(N - c + 1)$ . Мы не делаем каких-либо предположений относительно  $F$ , только лишь вычисляем эмпирическую функцию распределения  $\hat{F}(x)$  на основе наблюдений следующим образом.

На практике время постановки задачи в очередь, время начала и окончания работы дискретны, вычисляются в секундах. На интервале  $[0, T]$  каждую секунду измеряется количество занятых ресурсов. Это несложно сделать на основе данных  $\{(c_i, s_i, r_i), i = 1, \dots, n\}$ :

$$\xi(t) = \sum_{i=1}^n c_i \mathbb{I}_{t \in [s_i, s_i + r_i)}, \quad t \in 0, \dots, T,$$

тут  $\mathbb{I}$  означает индикатор. На основе выборки  $\{\xi(t), t \in \{0, \dots, T\}\}$  получаем эмпирическую функцию распределения:

$$\hat{F}(x) = \frac{1}{T+1} \sum_{t=0}^T \mathbb{I}_{\xi(t) < x}.$$

Индекс

$$W_3 = \frac{1}{n} \sum_{i=1}^n \frac{(s_i - q_i)}{r_i} \hat{F}(N - c + 1)$$

означает среднее время ожидания, взвешенное на время выполнения и вероятность мгновенного получения ресурсов для задачи данного размера.

На практике получение  $\hat{F}(x)$  для большого количества задач требует значительных вычислений. Упрощенно можно считать вероятность загрузки ресурсов равномерной на множестве  $\{0, \dots, N\}$  и рассматривать индекс

$$W_4 = \frac{1}{n(N+1)} \sum_{i=1}^n \frac{(s_i - q_i)(N+1 - c_i)}{r_i},$$

который вычисляется значительно проще.

Очевидно, что значения всех приведенных выше индексов зависят от кластера, на котором выполняются задачи, набора этих задач и планировщика, который их распределяет. Поэтому корректное сравнение планировщиков возможно лишь на одном кластере для одного набора задач, который должен быть достаточно большим и репрезентативным. В тестовый набор должны попадать основные типы задач по количеству узлов и времени выполнения, которые чаще всего запускаются пользователями данного кластера. Хорошие результаты сравнения дают также наборы реальных задач, которые запускались на кластере на протяжении значительного периода времени (несколько лет).

### 3. Тестовые результаты

В ходе эксперимента сравнивались два популярных менеджера ресурсов Torque [2] и Slurm [3], при этом Torque работал в режиме FIFO, а Slurm – в режимах Backfill и FIFO. Стенд моделировал работу кластера из 166 узлов и 1328 ядер.

Алгоритм планирования FIFO (First In First Out) обеспечивает принцип простой очереди, когда задачи получают ресурсы строго в той последовательности, в которой они появились в очереди. Этот алгоритм реализован в планировщиках Torque и Slurm.

Алгоритм планирования Backfill является развитием FIFO и поддерживается менеджером ресурсов Slurm. Особенность Slurm Backfill состоит в том, что он может без очереди пропускать задачи с низким приоритетом при условии, что это не задерживает выполнение задач с высоким приоритетом. Этот способ более эффективен в том случае, когда пользователи точно указывают ограничения на время выполнения задачи.

В качестве тестовой последовательности задач была выбрана статистика работы кластера Южно-Уральского университета за 10 месяцев. Для ускорения работы стенда использоваться коэффициент времени равный 20, что позволило сократить моделирование до двух недель. Моделирование сводилось к запуску на стенде задач, аналогичных оригинальным по количеству занятых узлов и времени выполнения.

Тестовая последовательность состояла из 32567 задач, реальная длительность которых превышала 20 секунд. Это требование связано с округлением моментов запуска и выполнения к ближайшей 20-секундной отметке. Параметры тестовой последовательности приведены в таблице 1.

Таблица 1. Параметры тестовой последовательности

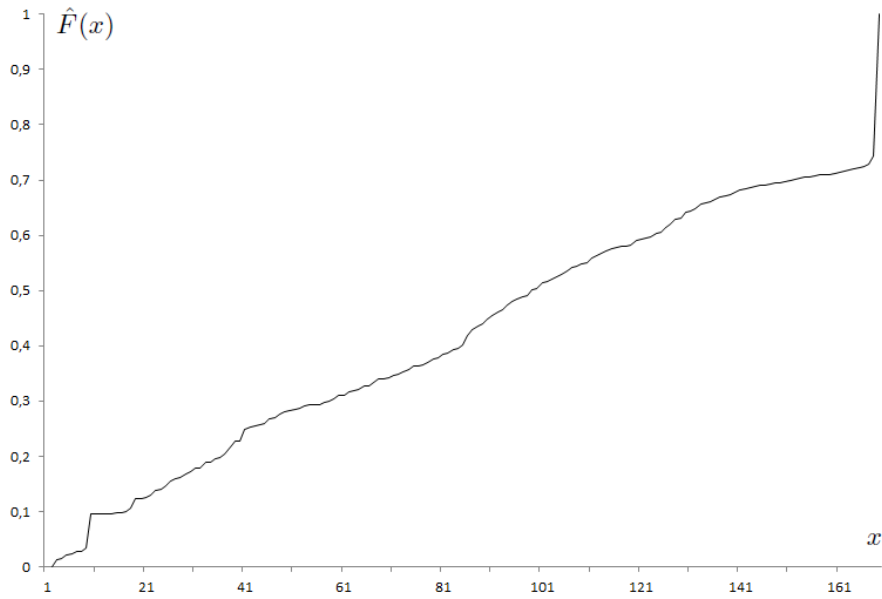
Среднее время работы задач, часов	12
Наименьшее количество узлов	1
Наибольшее количество узлов	100

На стенд по очереди устанавливались разные менеджеры ресурсов и алгоритмы планирования: Torque FIFO, Slurm Backfill и Slurm FIFO.

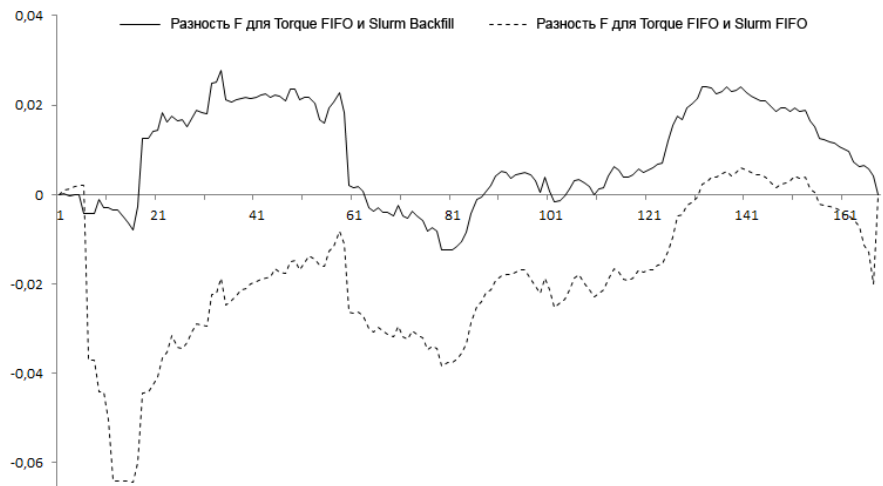
Для каждого из трех случаев на стенд загружалась тестовая последовательность задач таким образом, чтобы симитировать для планировщика моменты появления задач и необходимое количество ресурсов точно так же, как это происходит на реальном кластере. После запуска виртуальная задача "выполнялась" необходимое количество времени, после чего останавливалась. Стенд прекращал свою работу после завершения последней задачи в тестовой последовательности.

Журнал запусков задач обрабатывался, восстанавливался масштаб времени и проводилось вычисление всех вышеупомянутых индексов.

На рис. 1 изображена эмпирическая функция распределения занятых узлов для пла-



**Рис. 1.** График эмпирической функции распределения занятых узлов для Torque FIFO



**Рис. 2.** Разность эмпирических функций распределения

нировщика Torque FIFO. Она ведет себя подобно функции равномерного распределения, кроме последнего значения в точке  $x = 166$ , что соответствует полной загрузке кластера.

Эмпирические функции распределения занятых узлов для Slurm FIFO и Slurm Backfill незначительно отличаются от эмпирической функции распределения для Torque FIFO, (см. рис. 2). Отметим, что вероятность загрузки не более  $x$  узлов, где  $x \in [20; 40] \cup [110; 165]$ , для Slurm Backfill выше, чем у Torque FIFO. В случае сравнения эмпирических функций распределения Torque FIFO и Slurm FIFO видим, что, за некоторым исключением, соответствующая вероятность выше у Torque FIFO, чем у Slurm FIFO.

Результаты сравнения индексов для разных планировщиков приведены в таблицах 2 и 3. Цифры в таблице 2 округлены до целого для облегчения восприятия, цифры в скобках означают разность с Torque FIFO. Видим явное преимущество алгоритма Torque FIFO, который лучше ведет себя даже в сравнении с Slurm FIFO, что может указывать на лучшую реализацию алгоритма FIFO у планировщика Torque. Из таблицы 3 видим, что Torque FIFO выделяет ресурсы совсем по-другому, количество задач, которые получают ресурсы сразу, вдвое меньше этого же показателя у обеих версий Slurm. При этом Torque FIFO имеет

меньшее среднее время ожидания  $W$ .

**Таблица 2.** Сравнение индексов для разных планировщиков

	Torque FIFO	Slurm Backfill	Slurm FIFO
$W$	28597	31748 (+11%)	33035 (+16%)
$W_1$	83	101 (+21%)	105 (+26%)
$W_2$	351	417 (+19%)	434 (+23%)
$W_3$	62	74 (+21%)	80 (+29%)
$W_4$	83	100 (+21%)	104 (+26%)

**Таблица 3.** Процент задач, которые получили ресурсы сразу

Torque FIFO	Slurm Backfill	Slurm FIFO
19,56%	41,5%	41,05%

## 4. Выводы

Простроен программный стенд для моделирования работы кластера и проверки параметров взаимодействия программных систем, установленных на кластере, которые не требуют реальной вычислительной мощности. Стенд применен для практической проверки работы двух популярных менеджеров ресурсов и трех алгоритмов планирования: Torque FIFO, Slurm FIFO и Slurm Backfill. Введены и обоснованы несколько численных индексов для сравнения эффективности работы планировщиков в смысле времени ожидания предоставления ресурсов. По результатам сравнения однозначное преимущество имеет алгоритм Torque FIFO, поскольку в среднем он обеспечил более эффективное распределение ресурсов между задачами.

Сравнивая одинаковые алгоритмы планирования (FIFO), можно сказать, что Torque FIFO реализован лучше, поскольку в среднем обеспечивает на 11% меньшее время ожидания задач в очереди. Алгоритму Slurm Backfill удастся лучше справиться с увеличивающимся потоком задач, он быстрее предоставляет узлы новым задачам, обеспечивая наибольший процент задач с нулевым ожиданием.

Разработанная методика и построенный стенд позволяют подбирать планировщик, наиболее эффективный для каждого кластера. В частности, правильно подобранный планировщик позволяет повысить энергоэффективность кластера и эффективность прохождения вычислительных задач.

## Литература

1. Рейтинг Top500. URL: <http://top500.org> (дата обращения: 12.12.2011).
2. TORQUE Resource Manager. URL: <http://www.clusterresources.com> (дата обращения: 12.12.2011).
3. SLURM Resource Manager. URL: <https://computing.llnl.gov/linux/slurm/overview.html> (дата обращения: 12.12.2011).