Разработанный алгоритм проиллюстрирован на примере оценки уровня комплексной информационной безопасности компьютерной системы.

Литература

1. Ажмухамедов И.М. Математическая модель безопасности на основе экспертных суждений // ИКТ. Т.7, №4, 2009. – С. 103-107.
2. Рыжов А.П. Элементы теории нечетких множеств и измерения нечеткости. М.: Диалог-МГУ, 1998. – 102 с.
3. Kaufmann A., Gupta M. Introduction to Fuzzy Arithmetic: Theory and Applications. Van Nostrand Reinhold, 1991. – 161 p.
4. Поспелов Д.С. «Серые» и/или «черно-белые» [шкалы] // Прикладная эргономика. Спец. выпуск «Рефлексивные процессы». №1, 1994. – С.15-21.
5. Yager R. Families of OWA operators // Fuzzy Sets and Systems. 59, 1993. – P.53-59.
6. Проталинский О.М. Применение методов искусственного интеллекта при автоматизации технологических процессов. Астрахань: Изд. АсГТУ, 2004. – 184 с.

УДК 621.394.76

АНАЛИЗ АРХИТЕКТУР КЛАСТЕРОВ ДЛЯ СЕТЕВЫХ ЗАДАЧ

Уривский А.В., Чефранова А.О.

В статье проводится анализ архитектур построения кластеров для обработки и защиты информации, передаваемой по сетям связи. Строится теоретическая модель вычислений с учетом особенностей сетевой обработки и возможности реализации кластера из широкодоступных неспециализированных компонент. Архитектуры сравниваются по критериям производительности, отказоустойчивости и требованиям к пропускной способности сетей.

Введение

Один из способов организации высокопроизводительных вычислений состоит в параллельной обработке поступающего потока заданий на нескольких независимых элементах, образующих вычислительный кластер. Обычно элементами кластера (далее – ЭК) являются отдельные компьютеры или серверы, связанные одной или несколькими локальной коммуникационными сетями.

Регулированием состава кластера можно добиться заданной производительности. В мировом листинге суперкомпьютеров Топ-500 порядка 80% участников являются кластерами. Другой особенностью кластеров является обеспечение отказоустойчивости за счет естественного дублирования основных функций различными ЭК. Важным также является и то, что кластеры могут создаваться из уже имеющихся и широкодоступных стандартных компонент. В частности в качестве ЭК могут выступать устаревающие компоненты относительно низкой производительности. При этом характеристики кластера будут соответствовать современному уровню. Именно такого рода кластеры, создаваемые из неспециализированных компонент, являются предметом нашего исследования.

В основном кластеры используется для вычислительно трудоемких научных расчетов. Однако с интенсивным развитием инфокоммуникационных сетей высокие требования по производительности предъявляются и к сетевому оборудованию. Речь идет об исполнении разнообразных сетевых заданий: маршрутизации и транзитной обработке данных, защите информации и фильтрации трафика, почтовых службах, и т. п. Можно выделить некоторые особенности таких заданий. Во-первых, количество заданий, поступающих в единицу времени для обработки, весьма велико. Во-вторых, сложность отдельного задания относительно мала. В-третьих, обычно можно считать, что сложность задания линейно зависит от объема самого задания и/или результата исполнения. Указанные особенности существенно облегчают балансировку заданий по отдельным элементам кластера, но предъявляют специальные требования к организации их взаимодействия.

Кластер с точки зрения источника заданий и потребителя результатов представляется единым логическим сетевым элементом. При одной и той же физической организации, которая диктуется в основном используемой сетевой технологией, возможны различные логические архитектуры – способы организации взаимодействия множества ЭК. Наиболее важные архитектурные вопросы включают в себя маршрутизацию заданий и результатов обработки внутри кластера и балансировку (распределение) заданий по ЭК для обеспечения заданных характеристик. В этой статье мы рассмотрим, как влияет выбор архитектуры на

производительность кластера с учетом особенностей обработки сетевых заданий.

Сетевая структура кластера

Объекты, не входящие в состав кластера, будем относить к внешней сети.

На вход кластера из внешней сети поступает поток заданий. Результаты работы выводятся через выход кластера. В силу природы сетевой обработки любой реальный физический сетевой интерфейс кластера может функционировать и как вход, и как выход. При этом будем подразумевать, что у каждого ЭК имеется, как минимум, два физических сетевых интерфейса для соединения с внешней сетью.

Для организации эффективной работы кластера обычно создают отдельную служебную сеть, отделенную от внешней сети. К этой сети подключены все ЭК и по ней они обмениваются как данными при исполнении заданий, так и служебной информацией.

Схематически сетевая структура кластера изображена на рис. 1, где изображенные элементы сетевой инфраструктуры соответствуют выбранной технологии канального и физического уровня взаимодействия ЭК. Подразумевается, что это стандартные, свободно доступные устройства, коммутации, обеспечивающие информационный обмен в соответствии с адресной информацией.

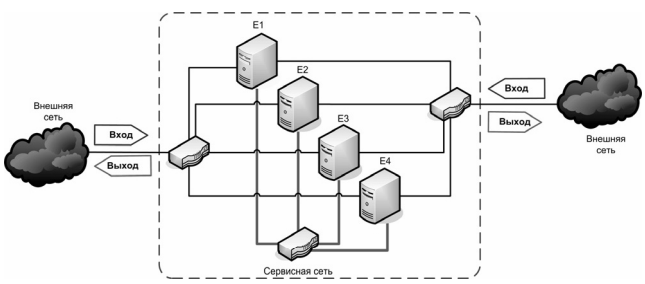

Рис. 1. Сетевая структура кластера

Параметры потока заданий и кластера

Задание – это некоторый набор данных, которые подаются на вход кластера для исполнения. Задание характеризуются размером описания v_{in} – числом двоичных разрядов в таком наборе данных. Исполнение задания приводит к некоторому результату, размер которого v_{out}.

Сложность задания w – это число операций, которое надо совершить для исполнения задания. Для разных заданий может требоваться выполнение различных операций. Мы будем полагать, что все разнородные операции приведены (пересчитаны) к одному типу, и во везде рассматриваются одни и те же операции.

Каждый ЭК характеризуется некоторой производительностью p_i, которая выражается числом операций, исполняемых за единицу времени. Кластер, как совокупность ЭК, характеризуется максимальной производительностью $P_{\max} = \sum_i p_i$, задающей предельные возможности кластера и не связанной с потоком заданий и архитектурой, и фактической производительностью кластера P – средним числом операций в единицу времени для данного потока заданий на заданном интервале времени. Для потока однотипных заданий производительность кластера можно характеризовать числом исполненных заданий J за заданный временной интервал.

Пропускная способность сетевого интерфейса есть максимальная скорость передачи информации через интерфейс, выраженная в двоичных единицах информации в единицу времени. Для пропускной способности c-го сетевого интерфейса i-го ЭК будем использовать обозначение $C(i, c)$. Для служебной сети будем использовать индекс s.

В рассматриваемой модели мы не учитываем латентность каналов передачи, что адекватно выбранной стратегии использования стандартных сетевых устройств. Ее влияние при обмене пакетами малого размера можно учесть косвенно через снижение эффективной пропускной способности.

Базовые архитектуры

Для достижения высокой производительности кластера необходимо обеспечить оптимальную загрузку каждого ЭК. Балансировка заданий и заключается в поиске их оптимального размещения по ЭК.

При централизованной балансировке решение о размещении принимает один выделенный ЭК, называемый балансировщиком или диспетчером. Альтернативой является распределенная балансировка, при которой каждый ЭК просматривает весь поток заданий и принимает решение исполнять ему данное задание или нет.

Для балансировки и исполнения задания, его нужно разместить на соответствующем ЭК.

При широковещательном размещении каждое задание одновременно подается на все ЭК.

В случае централизованной балансировки широковещательное размещение подразумевает, что задание помещается в очередь каждого ЭК и остается там до тех пор, пока балансировщик не

примет решение, какой ЭК это задание должен исполнить.

При централизованной балансировке возможно и адресное размещение заданий: задания поступают на вход балансировщика, который после принятия решения о размещении пересылает это задание на соответствующий ЭК.

Поскольку вход и выход кластера могут реализовываться через один и тот же сетевой интерфейс, то вывод результатов исполнения заданий аналогичен их размещению. При параллельном выводе результатов каждый ЭК самостоятельно отправляет результаты во внешнюю сеть. Вывод результатов исполнения может происходить и через один выделенный ЭК. В дальнейшем будем рассматривать вариант, когда ЭК, контролирующий выход, совпадает с балансировщиком.

Рассматривая комбинации процедур размещения заданий, их балансировки и вывода результатов, можно выделить 6 основных типов архитектур построения кластеров:

- полностью централизованная: адресное размещение заданий, централизованная балансировка и контролируемый вывод результатов;
- полностью распределенная: широковещательное размещение заданий, распределенная балансировка и параллельный вывод результатов;
- гибридные: четыре остальные комбинации (кроме нереализуемых комбинаций с адресным размещением заданий и распределенной балансировкой).

Безусловно, указанные архитектуры не покрывают всего многообразия возможностей при построении кластеров. Мы, например, сознательно не рассматриваем конвейеризации при исполнении заданий. Это связано с упоминавшейся особенностью сетевых заданий: сложность каждого из них низка, и любой ЭК может выполнить любое задание. Поэтому разбиение обработки на множество мелких этапов, очевидно, снизило бы эффективность обработки и, кроме того, усложнило бы анализ.

Далее мы проанализируем три кластерные архитектуры – полностью централизованную и распределенную и гибридную (с широковещательным размещением заданий, централизованной балансировкой и параллельным выводом результатов).

Рабочие характеристики кластера

Пусть кластер состоит из M ЭК с производительностями $(p_1,\ldots,p_M)$. Обозначим $p_{\min}=\min_i p_i$. Рассмотрим некоторый интервал времени T. Пусть за этот интервал кластер может обработать J общей размером описания заданий $V_{in}=\sum_j^J v_{in}(j)$,, и размером описания результатов $V_{out}=\sum_j^J v_{out}(j)$.

Характерной особенностью сетевых заданий, например шифрования и фильтрации трафика, является то, что трудоемкость их исполнения, как правило, линейно зависит от размеров описания задания и результата: $w(j)=\beta+\alpha v_{in}(j)+\gamma v_{out}(j)$. Аналогичной зависимостью описываются и трудоемкости вспомогательной обработки. Вспомогательная обработка включает предварительные операции (прием сетевых пакетов, их дефрагментация и распаковка для получения всего задания и т.д.), балансировку, операции вывода результатов, и внутрикластерные операции (переупаковку задания и пересылку его между ЭК). Размерность величин α и γ есть число операций на двоичный разряд описания, величины β – число операций.

Для операций, связанных с исполнением задания, у величин α,β и γ будем использовать верхний индекс *c*. При этом нижний индекс 0 будет соответствовать непосредственно исполнению задания, 1 – предварительной обработке, 2 – переупаковке и пересылке задания на другой ЭК, 3 – отправке результатов исполнения во внешнюю сеть, 4 – приему и переотправке результатов другим ЭК.

Для параметров, связанных с балансировкой (и относящихся к балансировщику), будем использовать верхний индекс *в*. При этом нижний индекс 1 соответствует процедуре балансировки, 2 – пересылке результата балансировки на другой ЭК, 3 – обработке результата балансировки на принимающем ЭК. Объем пересылаемых данных с результатами балансировки будем считать одинаковым для любого задания и равным v. Исходя из логики обработки данных, можно полагать, что $\alpha_3^c=\alpha_4^c=\alpha_2^b=\alpha_3^b=0$, $\gamma_1^c=\gamma_2^c=\gamma_1^b=\gamma_2^b=\gamma_3^b=0$.

При любой архитектуре балансировщик должен выполнять предварительную обработку, что требует $W_{in}=\alpha_1^c V_{in}+\beta_1^c J$ операций, и балансировку заданий – $W_{bal}=\alpha_1^b V_{in}+\beta_1^b J$. Кроме того, в любой архитектуре должны быть исполнены сами задания, что требует $W_c=\alpha_0^c V_{in}+\gamma_0^c V_{out}+\beta_0^c J$ операций, и проведен вывод результатов – еще $W_{out}=\gamma_3^c V_{out}+\beta_3^c J$ операций. Суммарную трудоемкость этих операций $W_A=W_{in}+W_{bal}+W_c+W_{out}$ можно считать архитектурно-независимой трудоемкостью обработки. Тогда величину $L=TP_{\max}-W_A$ можно рассматривать как архитектурно-зависимые потери производительности,

связанные с обменом информацией между ЭК или отсутствием такового. Минимизация L правильным выбором архитектуры максимизирует количество обрабатываемых заданий при прочих равных условиях.

Таблица 1. Требования к пропускным способностям

Архитектура	Пропускные способности
Централизованная	$V_{in} \le TC(b,in)$ $V_{out} \le TC(b,out)$ $V_{in}+V_{out} \le TC(b,s)$ $V_{in}+V_{out} \le T\sum_{i\backslash b} C(i,s)$
Гибридная	$V_{in} \le TC(i,in)$ $vJ \le TC(b,s)$ $vJ \le T\sum_{i\backslash b} C(i,s)$ $V_{out} \le T\sum_{i} C(i,out)$
Распределенная	$V_{in} \le TC(i,in)$ $V_{out} \le T\sum_{i} C(i,out)$

Рабочие характеристики кластера для трех архитектур сведены в таблицы 1-3. При составлении таблиц подразумевалось, что поток состоит из одинаковых заданий. Результаты естественным образом обобщаются на случай разнородных заданий суммированием по типам заданий. Также подразумевалась идеальная по эффективности балансировка.

Таблица 2. Нагрузка на балансировщик

Архитектура	Нагрузка на балансировщик
Централизованная	$Tp_b \ge W_{in}+W_{bal}+$ $+(\alpha_2^c V_{in}+\beta_2^c J)+(\gamma_4^c V_{out}+\beta_4^c J)$
Гибридная	$Tp_b \ge W_{in}+W_{bal}+\beta_2^b J$
Распределенная	$Tp_{\min} > W_{in}+W_{bal}$

Таблица 3. Архитектурно-зависимые потери

Архитектура	Потери
Централизованная	$(\alpha_2^c V_{in}+\beta_2^c J)+W_{in}+(\gamma_4^c V_{out}+\beta_4^c J)$
Гибридная	$W_{in}+(\beta_2^b+\beta_3^b)J$
Распределенная	$(M-1)(W_{in}+W_{bal})$

Из таблицы 1 видно, что с точки зрения размещения заданий централизованная архитектура наименее производительная: балансировщик по интерфейсу служебной сети должен пропустить через себя как полный поток заданий, так и результатов исполнения. Сетевая пропускная способность балансировщика и сервисной сети становятся критически важными показателями для этой архитектуры. Кроме того, среди всех архитектур в централизованной балансировщик оказывается самым загруженным, что следует из таблицы 2. Использование такой архитектуры, обеспечивающей максимальную управляемость кластера, оправдано, если балансировщик обладает существенно большей производительность, чем остальные ЭК, а сложность заданий такова, что пропускная способность не является ограничением.

Величины V_{in}, J, V_{out} и W (с разными индексами) растут примерно одинаково быстро с увеличением количества ЭК в кластере. Поэтому наибольшая скорость роста потерь, как видно из таблицы 3, наблюдается в распределенной архитектуре: рост примерно в M раз быстрее, чем в других.

Самой сбалансированной оказывается гибридная архитектура: потери производительности сравнимы с таковыми в централизованной архитектуре, а требования к пропускной способности и производительности балансировщика близки к таковым в распределенной архитектуре.

Результаты моделирования

Для проверки теоретической модели были проведены моделирующие тесты. Кластер полностью централизованной архитектуры, состоящий из одинаковых элементов, использовался для организации защищенного туннеля: данные, поступающие на вход, зашифровывались и передавались на выход с одновременной трансляцией сетевых адресов. В качестве данных использовался поток UDP-пакетов одинакового размера. Использование именно UDP протокола для передачи позволяло исключить возможное влияние источника и приемника пакетов, а одинаковый размер всех пакетов позволил упростить расчеты.

Можно считать, что каждое задание для кластера состояло в зашифровании блока данных из исходного пакета, помещении зашифрованных данных в новый пакет и подмене сетевых адресов. При такой обработке размеры исходного и обработанного пакетов практически совпадают, поэто-

му обозначим $\nu_{in} = \nu_{out} = \nu_0$. Исходя из типа заданий, будем считать, что $\beta_1^c = \beta_2^c = \beta_3^c = \beta_2^b = \beta_3^b = \beta$ и $\beta_4^c = \beta_2^c + \beta_3^c = 2\beta$, $\gamma_0^c = 0$. Для балансировки применялся простейший, но оптимальный для рассматриваемых условий, алгоритм round-robin. Поэтому сложностью балансировки в расчетах можно пренебречь, то есть $\alpha_1^b = 0$ и $\beta_1^b = 0$.

Нетрудно вычислить, что в случае отдельного элемента с производительностью p выполняется следующее соотношение для числа J_1 обработанных пакетов за интервал T: $Tp = [(\alpha_0^c + \alpha_1^c + \gamma_3^c)\nu_0 + 2\beta + \beta_0^c]J_1$.

Максимальная производительность в централизованной архитектуре достигается тогда, когда балансировщик осуществляет только балансировку, но не исполняет задания. Максимальное число пакетов, которое сможет обработать балансировщик за интервал T составит J_C: $Tp = [(\alpha_1^c + \alpha_2^c + \gamma_4^c)\nu_0 + 4\beta]J_C$. Максимальная производительность кластера по отношению к производительности одного элемента составит величину

$$\frac{J_C}{J_1} = \frac{(\alpha_0^c + \alpha_1^c + \gamma_3^c)\nu_0 + 2\beta + \beta_0^c}{(\alpha_1^c + \alpha_2^c + \gamma_4^c)\nu_0 + 4\beta}. \qquad (1)$$

В качестве прототипа для ЭК использовались компьютеры под управлением операционной системы семейства MS Windows с двумя сетевыми интерфейсами и установленным программным обеспечением ViPNet Coordinator [5]. Обработке подвергались UDP-пакеты размером 41 Кбайт. Такой размер позволить четко отделить основную обработку, которая занимала порядка 80% мощности ЭК от вспомогательных операций. В результате тестов была оценена относительная трудоемкость различных операций по обработке пакетов. В частности $\alpha_0^c \approx 77/15\alpha_1^c$, $\alpha_2^c \approx 5/15\alpha_1^c$, $\gamma_3^c \approx 6/15\alpha_1^c$, $\gamma_4^c \approx 7/15\alpha_1^c$, $\beta \approx \beta_0^c \approx 7/15\alpha_1^c \nu$. Подставляя оценки в формулу (1), получим $J_C / J_1 = 2{,}16$.

При практической реализации схемы была получена скорость туннелирования 190 Мбит/С на одном элементе, и максимум в 350 Мбит/С был достигнут при трех элементах в кластере. Таким образом, максимальное увеличение производительности составило $J_C^* / J_1 = 1{,}84$. Дальнейшее добавление элементов в кластер скорость обработки не увеличивало. Это говорит о том, что, как это и следовало из модели, ограничителем выступал балансировщик.

Заметим, что если бы моделировалась гибридная архитектура кластера, то число обработанных пакетов, при котором наблюдалась бы полная загрузка балансировщика, была бы выше почти в 2 раза.

В целом, с учетом принятых допущений, упрощений и погрешностей замеров, можно говорить о том, что модель довольно точно описывает поведение реального кластера как качественно – ограничение со стороны балансировщика, так и количественно – через достижимое увеличение скорости обработки.

Выводы

В исследовании проводится сравнительный анализ архитектур построения кластеров для решения сетевых задач. К такого рода задачам относятся задачи сетевого кодирования, маршрутизации, сетевой защиты информации, фильтрации трафика и т.п. Характерными особенностями таких задач являются высокая плотность потока заданий, невысокая трудоемкость решения каждого задания, линейно зависящая от размера описания задания и/или размера описания результата.

Были рассмотрены простые архитектуры построения кластера, которые относительно легко реализуются с использованием стандартных широкодоступных неспециализированных компонент. Архитектуры сравнивались по критериям сетевой пропускной способности, отказоустойчивости и производительности при обработке потока заданий.

Основной вывод исследования состоит в том, что наиболее сбалансированной архитектурой для построения кластера для решения сетевых задач является та, в которой балансировку осуществляет выделенный элемент кластера, производится широковещательное размещение заданий сразу на все элементы и вывод результатов исполнения производится тем элементом, который исполнял задание.

Литература

1. Гергель В.П. Теория и практика параллельных вычислений. М.: Бином. Лаборатория знаний, 2007. – 423 с.
2. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. СПб.: БХВ-Петербург, 2002. – 608 с.
3. CheckPoint Cluster XL. Administration Guide. 2007. – 223 p.
4. ViPNet Coordinator. Руководство администратора. Версия 3.0. ОАО «ИнфоТеКС». Москва, 2009. – 22 с.