5. *Берилло, А.* Nvidia CUDA – неграфические вычисления на графических процессорах [Электронный ресурс] / IXBT. – 2016. – Режим доступа: http://www.ixbt.com/video3/cuda-1.shtml. – Дата доступа: 16.03.2016
6. MATLAB Documentation [Электронный ресурс] / Mathworks – 2016. – Режим доступа: http://www.mathworks.com/help/index.html. – Дата доступа: 03.02.2016

РАСПРЕДЕЛЕННЫЕ СИСТЕМЫ ОБРАБОТКИ ДАННЫХ НА ОСНОВЕ ВЕБ-СЕРВИСОВ

Н. С. Левкович

ПАРАЛЛЕЛЬНОЕ ПРОГРАММИРОВАНИЕ НАЧИНАЕТСЯ СО СКРЫТЫХ ОШИБОК

Параллельные информационные технологии в наше время занимают должную нишу знаний разработчика современного ПО, поэтому мы не могли не затронуть эту интересную тему. В процессе создания программ с применением параллелизма разработчик сталкивается с неучтенными сложностями, которые в дальнейшем приводят к появлению дополнительных, по сравнению с последовательными архитектурными решениями, ошибок. Основываясь на опыте и применении можно выделить список стандартных ошибок, возникающих при разработке, и, зная это, можно в дальнейшем обойти эти проблемы, предугадав слабые места в программе. Логика программы [1] – набор свойств, выраженных в форме логических утверждений (предикатов), используемых программистом при построении программы. Известно, что эти заявления следуют из математического решения задачи и должны быть истинны в поставленных целевых моментах программы. Формально, можно высказать следующее относительно логики параллельных программ:

1. Претендует на всеохватность.
2. Сложна.
3. Имеет “продвинутый” уровень.

Также нельзя не затронуть профилирование. Факторы, от которых зависит производительность параллельных потоков: объем данных, структура исходных данных, упаковка и число ядер [2]. За какое время выполнится задача при параллельной работе потоков на разном количестве доступных ядер для использования во время выполнения?

Закон Амдала – иллюстрирует ограничение роста производительности вычислительной системы с увеличением количества вычислений. Согласно этому закону, ускорение выполнения программы за счёт распараллеливания её инструкций на множестве вычислителей ограни-

чено временем, необходимым для выполнения её последовательных инструкций. Теперь мы можем теоретически, применяя формулу по закону Амдала, рассчитать ускорение выполнения программы за счет распараллеливания [3].

Ускорение рассчитываются по формуле (1.1):

$$S = 1/(a + (1 - a)/p), \quad (1.1)$$

где p – количество ядер;
a – доля последовательного кода.

ОБЩАЯ МЕТОДИКА ПОСТРОЕНИЯ РАСПРЕДЕЛЕННЫХ СИСТЕМ. СПОСОБЫ БАЛАНСИРОВКИ НАГРУЗКИ МЕЖДУ НОДАМИ ВЕБ-СЕРВЕРА

Резюмируя вышесказанное, можно выделить базовый вопрос:

Что же такое распределенная система и,что в этом хорошего?

В литературе можно найти различные определения распределенных систем, причем ни одно из них не является удовлетворительным и не согласуется с остальными. Для наших задач хватит достаточно вольной характеристики.

Распределенная система – это набор независимых компьютеров, представляющийся их пользователям единой объединенной системой [4]. Эмпирические свойства:

1. Устойчивость при отказе одного из элементов;
2. Неоднородность элементов, конфигурация системы может изменяться в процессе функционирования;
3. Ни один из элементов не обладает полными знаниями обо всей системе в целом;

Представлены результаты исследования распределенных систем, а также предложены актуальные решения создания распределенной системы веб-приложений и сервисов на базе следующих архитектур:

1. кластерная архитектура;
2. клиент-сервер;
3. Peer - 2 - Peer;

Идея кластерных архитектурных решений состоит в организации совокупности сервисов для высоконагруженных приложений. Кластеры распределения нагрузки (Network Load Balancing, NLB) [5] – кластеры, распределяющие запросы через один или несколько входных узлов, которые перенаправляют их на обработку в остальные, вычислительные узлы, вследствие этого происходит: обеспечение высокой доступности сайта, его масштабирование в условиях возрастающей нагрузки и балансировка.

Проанализировав характеристики, а также функциональные возможности была разработана классификация кластерных архитектур, исследованы вариации реализации распределенных систем на базе кластерной архитектуры. Выделено три группы кластеров по функциональным особенностям: “Высокоскоростные”, “Системы Высокой Готовности”, “Смешанные Системы”.

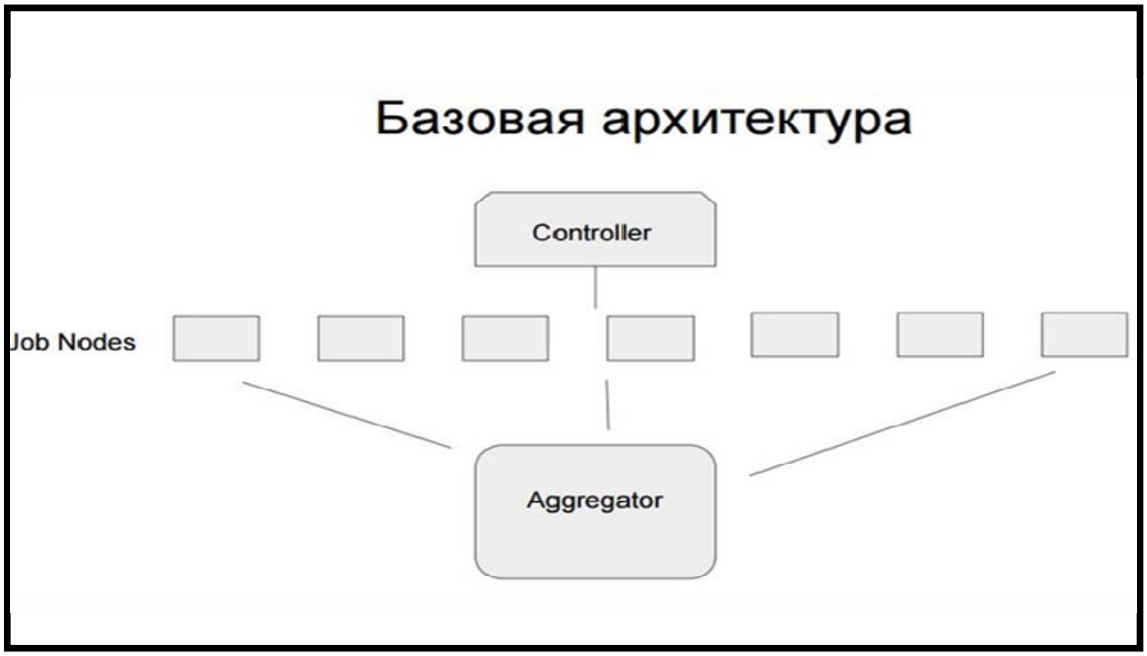

Рис.1. MapReduce

Описание: в систему прилетает Task → Task разбивается на маленькие Job → Каждая Job мапится на соответствующего воркера → Каждый воркер выполнения Job отсылает результаты в аггрегатор → Аггрегатор дожидается выполнения всех Job и отдает итоговый результат.

Литература

1. *Shelekhov V. Tumurov E*. Logic of non-interacting programs and reactive systems // Vestnik of Buryat State University. Section: mathematics, informatics. Issue 9/2012. – Ulan-Ude, 2012. 81–90 с.
2. *Уорбэртон Р*. Лямбда-выражения в Java 8. Функциональное программирование – в массы // ДМК Пресс, 2014. 192 с.
3. Интернет-адрес: https://ru.wikipedia.org/wiki/Закон_Амдала
4. *Э. Таненбаум, М. Ван Стеен.* Распределенные системы. Принципы и парадигмы // СПб.: Питер, 2003. 877 с.
5. Интернет-адрес: https://ru.wikipedia.org/wiki/Кластер_(группа_компьютеров)