Система распределенных высоконагруженных вычислений Aurora

Повар А.В.[1], Прудников П.В.[1]

[1]Кафедра теоретической физики, Омский государственный университет им. Ф.М. Достоевского, Омск, Россия

a.v.povar@gmail.com, prudnikp@univer.omsk.su

Аннотация. Значительное количество технологий и интерфейсов для параллельных вычислений сильно усложняют задачи разработки, отладки и запуска комплексных задач на кластерных и грид системах. Зачастую правильная организация взаимодействия программных компонент в конкурентной среде в ходе распределенных вычислений оказывается не менее сложной задачей, чем сам расчет физической модели. Система распределенных высоконагруженных вычислений Aurora призвана упростить проведение масштабных параллельных вычислений, предоставляя высокоуровневые интерфейсы, консолидирующие различные протоколы и технологии передачи данных, системы хранения и анализа данных, а также среду исполнения на основе JVM, позволяющую запускать приложения на различных платформах, и значительно упростить отладку и разработку прикладного программного обеспечения. Тестовыми платформами являются кластер СКИФ МГУ Чебышев и собственная грид система.

Ключевые слова

Распределенные вычисления, технология MPI, кластерые вычислительные системы, грид, JVM, Java, Scala.

1 Введение

В настоящее время подавляющее большинство кластерных суперкомпьютерных систем предоставляет технологию MPI[1] для взаимодействия с прикладным программным обеспечением. Однако, в ряде случаев интерфейс, предоставляемый MPI является не достаточно эффективным. В первую очередь это относится не к производительности самой системы, которая в достаточной мере высока (латентность на уровне MPI 1.3 - 1.95 мкс, скорость обмена данными порядка 1540 Мбайт/с для СКИФ МГУ Чебышев)[2], а к производительности разработчика, который вынужден заниматься не только проблемами, напрямую связанными с моделированием, но и сопутствующими проблемами интеграции с системой распределенных вычислений, которые, не смотря на стандарты могут в значительной мере отличаться друг от друга.

Не смотря на то, что MPI - это интерфейс обмена сообщениями, полноценный обмен сообщениями и реализация модели акторов[3] является достаточно сложной задачей, отчасти в силу того, что в большинстве случае используется связка MPI и C/C++, которая требует высокой квалификации разработчика и постоянного внимания ко всем деталям, начиная от управления памятью и заканчивая работой с сетевыми базами данных, а также из-за не самого простого интерфейса работы с MPI.

Стоит также отметить, что кроме узкоспециализированных кластерных систем существует еще целый класс вычислительных грид-систем, являющихся значительно более доступными для построения и использования. Такие системы обычно представляют собой группу недорогих компьютеров, соединенных в сеть посредством Ethernet. Значимость таких систем ни в коей мере нельзя недооценивать. Подобные системы находят очень широкое применение: начиная от обучения и организации небольших вычислительных платформ для исследовательских лабораторий, вплоть до известного проекта BOINC[4], с наиболее ярким представителем LHC@home[5]. Однако, в подавляющем большинстве случаев программное обеспечение, разработанное для интерфейса MPI не может быть запущено на подобных платформах без значительных усилий.

Таким образом, возникает необходимость создания некоторого промежуточного слоя абстракции, который бы предоставил интерфейс обмена сообщениями, утилизирующий существующие коммуникационные протоколы в качестве транспорта данных, скрывая различную их природу.

Кроме того, возникает ряд смежных проблем, не относящийся напрямую к протоколам взаимодействия. В первую очередь это задача управления данными научных вычислений, что включает в себя весь цикл, начиная от загрузки исходных данных, и их распределения, заканчивая сбором результатов и последующим их анализом. Разумеется не существует одного универсального решения для всех случаев, однако есть возможность максимально упростить наиболее типовые задачи по работе с данными и предоставить удобные механизмы взаимодействия с ними.

2 Система Aurora

Aurora представляет собой программный каркас и среду исполнения, основанную на модели акторов, с возможностью работы в гетерогенных сетях и предоставляющих сервисы обмена сообщениями, доступа к данным, управление выполнением, балансировки и перераспределения нагрузки.

Система реализована на платформе JVM с помощью языков программирования scala[6] и java. Поддерживается JNI[7] и механизмы вызова платформенно-зависимых программ и утилит. В том числе планируется возможность запуска кода для GPU.

Архитектурно система построена слоями. Наиболее специфичным является самый нижний - нулевой слой, предоставляющий механизм абстракции над сетевым взаимодействием. Сюда входит построение виртуальной сети с обособленой адресацией и маршрутизацией с использованием транспорта существующих сети (в первую очередь IP-сети и MPI протокол). Слои верхнего уровня предоставляют прикладному программному обеспечению механизмы взаимодействия с базами данных, системы запуска задач и приложений, инструменты управления и мониторинга.

Приложения могут быть написаны на любых JVM-based языках, сохраняющих совместимость с языком java. Это позволяет использовать более высокоуровневые языковые и инструментальные средства, увеличивая тем самым скорость разработки, возможно в ущерб скорости работы приложения. Также возможна интеграция посредством JNI с платформенно-ориентированным кодом(C/C++), что позволяет использовать весь арсенал доступных для платформы библиотек и утилит.

3 Заключение

В настоящий момент система проходит тестовые испытания на кластере СКИФ МГУ Чебышев.

Список литературы

[1] http://www.mpi-forum.org *Message Passing Interface Forum*

[2] http://parallel.ru/cluster/skif_msu.html *Суперкомпьютер СКИФ МГУ «ЧЕБЫШЕВ»*

[3] W. D. Clinger: Foundations of Actor Semantics. *AI Technical Reports*, AITR-633, 1981.

[4] http://boinc.berkeley.edu/ *Berkeley Open Infrastructure for Network Computing*

[5] http://lhcathome.web.cern.ch/LHCathome *LHC @ home*

[6] http://www.scala-lang.org/docu/files/ScalaReference.pdf *Scala programming language*

[7] http://docs.oracle.com/javase/6/docs/technotes/guides/jni/spec/jniTOC.html *Java Native Interface Specifications*