Московский государственный университет имени М.В.Ломоносова
Факультет Вычислительной математики и кибернетики

СУПЕРКОМПЬЮТЕРЫ И ПАРАЛЛЕЛЬНАЯ ОБРАБОТКА ДАННЫХ (лекция 8)

А.С.Антонов

Вед. н.с. НИВЦ МГУ, к.ф.-м.н.

asa@parallel.ru

ВМК МГУ, 2024

Методы оценки производительности

Производительность?

“Twelve Ways to Fool the Masses When Giving Performance Results on Parallel Computers”
David H. Bailey
Technical Report RNR-91-020, 1991

- *Вычислительное ядро: A = B+C*

- *Производительность: ‘ X ’ Mflops на Cray T90*

(например)

- *Пользователь: Я хочу больше, чем ‘X’...*

Производительность?

- *Вычислительное ядро: A = B+C*1*
- *Производительность: ‘2*X ’ Mflops на Cray T90 :))*

  А время осталось неизменным!!! :((

- *Пользователь: Я хочу больше, чем ‘X’...*
- *System Guru: Хочешь больше Mflops… Нет проблем...*

Методы оценки производительности

Какой компьютер выбрать?

В идеале было бы компьютеру однозначно сопоставить некое число.

***Пиковая производительность**: вычисляется просто и однозначно, но нет связи с реальной задачей пользователя. Даёт нижнюю оценку времени выполнения программы.*

Полезнее для пользователя оценка эффективности программно-аппаратной среды на некоторых задачах или наборе задач.

***Тест/бенчмарк (benchmark)** – программа или набор программ, используемые для определения сравнительных характеристик производительности вычислительной системы.*

Методы оценки производительности

Разновидности тестов (бенчмарков):

- ***Вычислительные ядра (kernels)*** *– небольшие ключевые части реальных приложений;*

- ***Игрушечные программы (toy programs)*** *– небольшие программы порядка 100 строк, как правило, решающие какую-то известную задачу, такую как быстрая сортировка;*

- ***Синтетические тесты (synthetic benchmarks)*** *– искусственно сконструированные программы, специально написанные, чтобы выполнять нагрузочное тестирование вычислительных систем или их компонент;*

- ***Микробенчмарк (microbenchmark)*** *- совсем небольшой тест, направленный на определение одной конкретной характеристики аппаратуры.*

Методы оценки производительности

Разновидности тестов (бенчмарков):

•*Иногда для тестирования вычислительных систем используют и* ***готовые приложения (application benchmarks),*** *взятые из той или иной области.*

•*Если такое приложение немного модифицируют для целей тестирования, то иногда используют термин* ***псевдо-приложение (pseudo-application).***

•*Любой из этих подходов не может вполне удовлетворять всем требованиям, предъявляемым к бенчмаркам. Поэтому значительное распространение получают* ***пакеты тестов (benchmarks suites),*** *в которых объединяют различные бенчмарки, характеризующие программно-аппаратную среду с разных точек зрения.*

Методы оценки производительности

Основные требования к тестам производительности (бенчмаркам):

- ***понятность*** *– должно быть ясно, какие аспекты системы тестируются, а результаты должны быть непротиворечивыми, лаконичными и легкими для понимания;*
- ***легкость в использовании****;*
- ***масштабируемость*** *– бенчмарк должен быть применим для различных по вычислительной мощности систем;*
- ***переносимость*** *– код должен быть легко адаптируем под любую вычислительную платформу;*
- ***репрезентативность*** *– бенчмарк должен представлять алгоритм из какой-то реальной области;*
- ***доступность*** *– должна быть возможность легко получить нужную версию бенчмарка;*
- ***воспроизводимость*** *– несколько запусков бенчмарка в одних условиях должны давать одинаковые результаты.*

Методы оценки производительности

Наиболее известные тесты (бенчмарки):

- *Linpack*
- *STREAM*
- *Ливерморские циклы*
- *Perfect Club Benchmarks*
- *SPEC*
- *HINT*
- *NAS Parallel Benchmarks (NPB)*
- *HPC Challenge*
- *Graph500*
- *HPCG*

Методы оценки производительности

Тест Linpack

- *Создан Джеком Донгаррой (Jack Dongarra) и его коллегами в 1979 году.*
- *Используется для формирования списков Top500 и Топ50.*
- *Решение больших систем линейных алгебраических уравнений Ax=b с плотной квадратной матрицей методом LU-разложения с выбором главного элемента по строке.*
- *Число операций с плавающей точкой оценивается по формуле $2n^3/3 + 2n^2$, где n – линейный размер матрицы.*
- *Сначала Linpack 100×100, запрет изменений текста.*
- *Далее - Linpack 1000×1000, стандартная головная часть программы.*
- *Сейчас – произвольный (максимально возможный) размер матрицы, возможность вносить любые изменения в текст.*

Методы оценки производительности

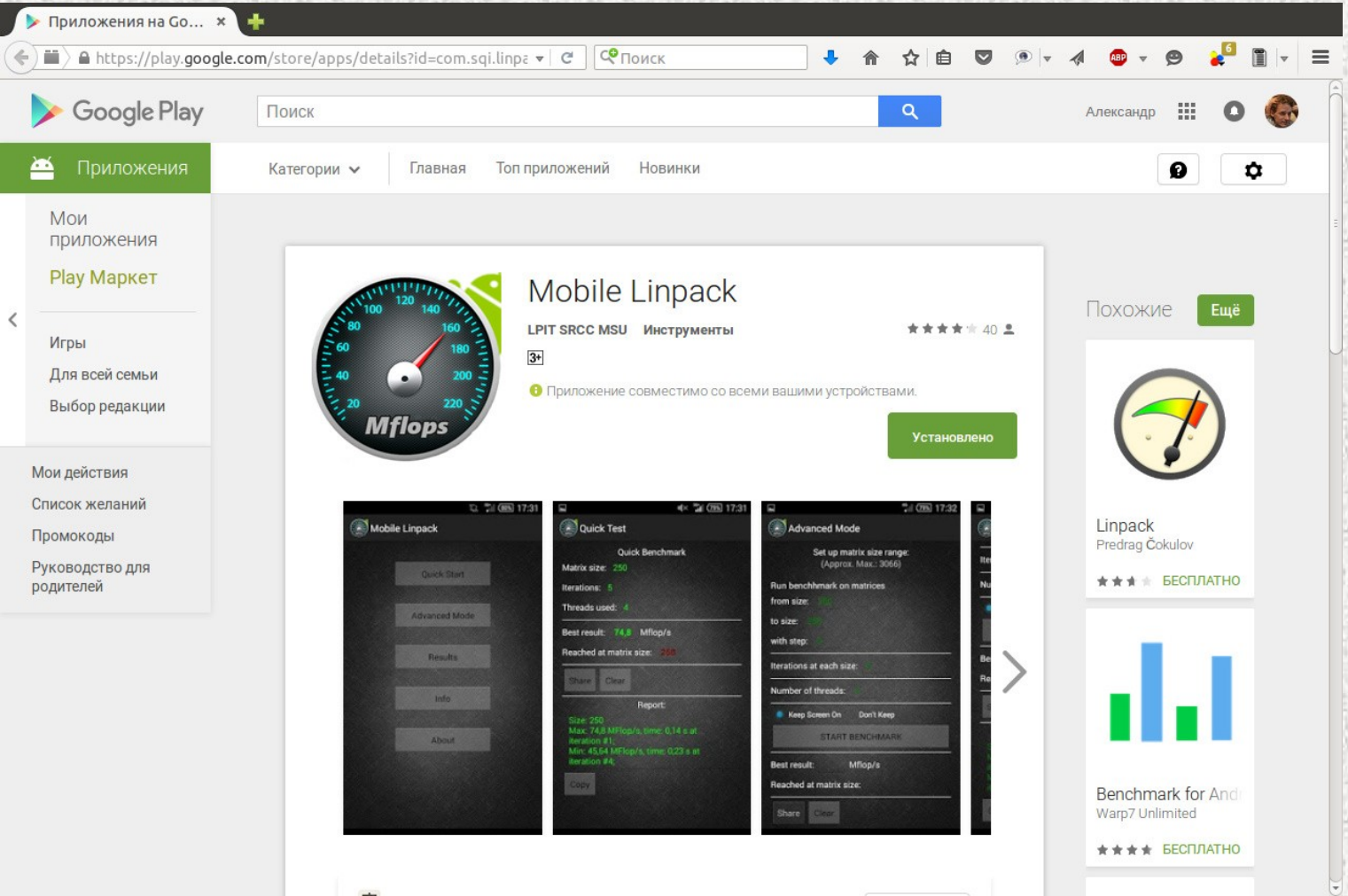
High Performance Linpack (HPL)

- *http://www.netlib.org/benchmark/hpl/*
- *Наиболее популярная реализация теста Linpack на языке Си.*
- *Обмены между процессорами выполняются через процедуры MPI.*
- *Вычисления на каждом процессоре основываются на низкоуровневой библиотеке базовых функций линейной алгебры BLAS (Basic Linear Algebra Subprograms) или VSIPL(Vector Signal Image Processing Library).*
- *Варианты BLAS:*
  - *ACML (AMD Core Math Library) для процессоров AMD Athlon и Opteron.*
  - *Intel MKL (Intel Math Kernel Library) для процессоров Intel.*
  - *cuBLAS в составе NVIDIA CUDA SDK для видеокарт и ускорителей.*
  - *ESSL (Engineering and Scientific Subroutine Library) для процессоров PowerPC.*
  - *ATLAS (Automatically Tuned Linear Algebra Software), реализация интерфейса BLAS с открытым исходным кодом.*
  - *uBLAS, часть библиотеки Boost.*

Методы оценки производительности

http://mobile-hpl.parallel.ru/

Реализация теста Linpack для мобильных устройств.
Версии для Android 1.6 и выше и для iOS 6.0 и выше.



Адрес для отзывов: ml@parallel.ru

Методы оценки производительности

STREAM (Sustainable Memory Bandwidth in High Performance Computers)

http://www.cs.virginia.edu/stream/

- *Синтетический тест, оценивающий скорость работы с памятью с простой арифметикой и без.*
- *STREAM. Производительность на операциях:*
  - *a(i)=b(i);*
  - *a(i)=q*b(i);*
  - *a(i)=b(i)+c(i);*
  - *a(i)=b(i)+q*c(i);*
  - *замер скорости передачи данных.*
- *STREAM2. Производительность на операциях:*
  - *a(i)=q;*
  - *a(i)=b(i);*
  - *a(i)=a(i)+q*b(i);*
  - *sum = sum + a(i);*

Методы оценки производительности

STREAM (Sustainable Memory Bandwidth in High Performance Computers)

- *Размер массивов задаётся достаточным, чтобы выйти за размеры кэш-памяти.*
- *Отношение пиковой производительности к скорости передачи данных почти всегда больше 1.Чем это отношение больше, тем больше* ***несбалансированность*** *компьютера.*
- *Параллельные версии с использованием OpenMP, pthreads и MPI.*

Простота и переносимость тестов вызывает недоверие, появляются более сложные тесты, что вызывает проблемы с переносимостью и т.д. Поэтому создаются наборы тестов.

Методы оценки производительности

NAS Parallel Benchmarks

- *http://www.nas.nasa.gov/Software/NPB/*
- *Набор тестов производительности, разработанных в NASA Advanced Supercomputing (NAS) Division (ранее NASA Numerical Aerodynamic Simulation Program).*
- *Существуют последовательная реализация, параллельные реализации с использованием MPI, OpenMP, MPI+OpenMP , вариант на JAVA, версия для Grid на базе Globus.*
- *Последняя на данный момент версия NPB 3.4.2.*
- *Размер задачи – классы:*
  - *S, W (для тестовых прогонов);*
  - *A, B, C – стандартные тесты (каждый в среднем в 4 раза больше предыдущего);*
  - *D, E, F – большие тесты (каждый в среднем в 16 раз больше предыдущего)*

Методы оценки производительности

NAS Parallel Benchmarks

- *5 вычислительных ядер:*
  - *IS (Integer Sort) – параллельная сортировка N целых чисел.;*
  - *EP (Embarrassingly Parallel) – генерация независимых нормально распределённых случайных величин при помощи полярного метода Марсальи;*
  - *CG (Conjugate Gradient) – приближение к наименьшему собственному значению большой разреженной симметричной положительно-определённой матрицы с использованием обратной итерации вместе с методом сопряжённых градиентов в качестве подпрограммы для решения СЛАУ;*
  - *MG (MultiGrid) – аппроксимация решения трёхмерного дискретного уравнения Пуассона при помощи V-циклового многосеточного метода;*
  - *FT – решение трёхмерного уравнения в частных производных при помощи быстрого преобразования Фурье.*

Методы оценки производительности

NAS Parallel Benchmarks

- *3 модельных приложения:*
  - *BT (Block Tri-diagonal solver) – блочная трёхдиагональная схема с методом переменных направлений;*
  - *SP (Scalar Penta-diagonal solver) – скалярная пятидиагональная схема;*
  - *LU (Lower-Upper Gauss-Seidel solver) – метод симметричной последовательной верхней релаксации (алгоритм SSOR, задача LU).*
- *3 дополнительных теста:*
  - *UA (Unstructured Adaptive mesh) – решение уравнения теплопроводности с учётом диффузии и конвекции в кубе;*
  - *DC (Data Cube) – оператор «куб данных»;*
  - *DT (Data Traffic) – симуляция обменов данными между узлами-источниками, узлами-обработчиками и узлами-потребителями.*

Методы оценки производительности

***HPC Challenge** - набор бенчмарков, который измеряет производительность вычислительной системы с разных точек зрения.*

- *http://www.hpcchallenge.org/*
- *Включает в себя 7 тестов:*
  - *HPL (Linpack);*
  - *DGEMM – вычисляет производительность перемножения матриц);*
  - *STREAM;*
  - *PTRANS – параллельное транспонирование матрицы;*
  - *RandomAccess – измеряет скорость целочисленных случайных обновлений памяти (GUPS);*
  - *FFT – реализация одномерного дискретного преобразования Фурье;*
  - *Communication bandwidth and latency – скорость передачи данных и латентность.*

Методы оценки производительности

Graph500

- *http://www.graph500.org/*
- *Анонсирован в 2010 году, составляется свой список наиболее производительных компьютеров.*
- *На данный момент последняя версия теста 3.0.0.*
- *Включает последовательный вариант, варианты на OpenMP, MPI и GNU Octave.*
- *Реализует поиск в ширину (BFS) и поиск кратчайшего пути от одной вершины (SSSP) в большом ненаправленном графе.*
- *Используется для измерения пропускной способности сетевой системы суперкомпьютеров.*
- *Производительность алгоритмов измеряется количеством пройденных дуг графа в секунду (Traversed Edges Per Second, TEPS). Используются обозначения ME/s и GE/s – миллионы и миллиарды пройденных дуг в секунду соответственно.*

Методы оценки производительности

HPCG

- *http://hpcg-benchmark.org/*
- *High Performance Conjugate Gradient – решение системы линейных алгебраических уравнений с разрежённой квадратной положительно определённой симметричной матрицей.*
- *Оценивает не только скорость самих вычислений, но и нерегулярных обращений в память.*
- *Разработчики: Jack Dongarra, Michael Heroux, Piotr Luszczek.*
- *Анонсирован в 2013 году.*
- *Реализация написана на C++ с поддержкой MPI и OpenMP.*
- *На данный момент последняя версия теста 3.1.*

Методы оценки производительности

Метрики производительности:

- ***MIPS (Million Instructions Per Second)*** *характеризует скорость выполнения компьютером своих инструкций.*
- ***MFLOPS (Million FLoating point Operations Per Second)*** *характеризует скорость выполнения компьютером арифметических операций над вещественными числами, представленными в форме с плавающей запятой.*
- ***Ускорение*** *показывает сравнение времени выполнения параллельной реализации с другими вариантами:*
  - ***абсолютное*** *– сравнение идёт со временем выполнения быстрейшей последовательной реализации на быстрейшем последовательном компьютере;*
  - ***реальное*** *– сравнение идёт со временем выполнения быстрейшей последовательной реализации на одном процессоре параллельного компьютера;*
  - ***относительное*** *– сравнение идёт со временем выполнения параллельной реализации на одном процессоре параллельного компьютера.*

Методы оценки производительности

Метрики производительности:

- ***Эффективность реализации*** *– отношение достигнутой производительности к пиковой производительности данного компьютера.*
- ***Эффективность распараллеливания*** *– отношение полученного ускорения к числу процессоров.*
- ***TEPS (Traversed Edges Per Second)*** *– применяется для графовых алгоритмов. Для получения результата в данной метрике необходимо разделить количество пройденных вершин графа на затраченное время.*
- ***GB/s (Gigabytes Per Second)*** *– пропускная способность коммуникационной сети. Размер данных, использованных в алгоритме, делится на время пересылки.*
- ***GUPS*** *скорость целочисленных случайных обновлений памяти.*
- ***MFLOPS/Watt*** *характеризует энергоэффективность выполнения некоторого алгоритма на конкретном компьютере.*

Методы оценки производительности

Необходимость комплексного тестирования программно-аппаратной среды

- *базовый уровень ПО (ОС, компилятор, системы программирования);*
- *базовый уровень аппаратуры (элементарные операции, иерархия памяти);*
- *уровень операций ввода/вывода;*
- *базовый коммуникационный уровень;*
- *коммуникационный уровень приложений;*
- *уровень модельных приложений;*
- *уровень реальных приложений.*

Технологии параллельного программирования

Технологии параллельного программирования

Модели параллельного программирования:

• ***SPMD*** *(**Single Program, Multiple Data**, единая программа, множество данных) – на всех процессорах выполняется одна и та же программа над своим множеством входных данных.*

• ***Master-Workers*** *или **Master-Slave** (мастер-рабочие) – один из процессов (называемый главным, master) получает входные данные задачи, разделяет их на части и передаёт другим процессам (рабочим, workers) для обработки, а затем получает результаты и проводит финальную обработку.*

• ***Message passing*** *(модель передачи сообщений). Программа порождает несколько задач с уникальными идентификаторами. Взаимодействие осуществляется посредством отправки и приема сообщений. Новые задачи могут создаваться во время выполнения параллельной программы, несколько задач могут выполняться на одном процессоре.*

Технологии параллельного программирования

Модели параллельного программирования:

• ***Data parallel*** *(модель параллелизма данных). Одна операция применяется к множеству элементов структуры данных. Программа содержит последовательность таких операций. Распределяемыми данными обычно являются массивы. Как правило, языки программирования, поддерживающие данную модель, допускают операции над массивами, позволяют использовать в выражениях целые массивы, вырезки из массивов. Каждый процессор отвечает за то подмножество элементов массива, которое расположено в его локальной памяти.*

•***Shared memory*** *(модель общей памяти). Задачи обращаются к общей памяти, имея общее адресное пространство. Требуется разграничение доступа к общим данным.*

Технологии параллельного программирования

Критерии выбора

Различных технологий много. На чём основываться при выборе наиболее подходящей?

- *Возможность создания эффективных программ.*
- *Возможность быстрого создания параллельных программ (продуктивность программирования).*
- *Переносимость программ, гарантии сохранения эффективности.*
- *Стоимость.*

Технологии параллельного программирования

При программировании систем с общей и распределённой памятью нужно учитывать разные факторы.

Компьютеры с общей памятью:

- *параллелизм вычислений;*
- *синхронизация доступа к общим данным.*

Компьютеры с распределённой памятью:

- *параллелизм вычислений;*
- *распределение данных;*
- *согласование параллелизма вычислений и распределения данных;*
- *организация пересылок данных.*

Технологии параллельного программирования

0. Распараллеливающий компилятор

Идея состоит в том, что компилятор достаточно интеллектуальный для того, чтобы мог сам извлекать параллелизм программ и отображать на архитектуру целевого параллельного компьютера.

Технологии параллельного программирования

1. Спецкомментарии

- *В чистом виде - компиляторы CRAY (CDIR$ NODEPCHK).*

- ***OpenMP** – стандарт для программирования на масштабируемых SMP-системах в модели общей памяти. Кроме спецкомментариев есть и языковые конструкции (вызов функций).*

- ***OpenACC** – стандарт, описывающий набор директив для написания гетерогенных программ, задействующих как центральный, так и графический процессор. Последняя версия стандарта – OpenACC 2.7. https://www.openacc.org*

Технологии параллельного программирования

2. Расширения существующих языков программирования

HPF (High Performance Fortran)*: ориентирован на создание переносимых программ. Отображение массив - массив-шаблон - виртуальный процессорный массив - физические процессоры. Есть и спецкомментарии, и языковые конструкции, например, оператор FORALL для параллельных циклов. Необходимые коммуникации и синхронизации реализуются компилятором. Сложность конструкций HPF оказалась непреодолимым препятствием для создания по-настоящему эффективных компиляторов. Есть система HPF Adaptor, переводящая программу с HPF на Fortran с MPI.*

Технологии параллельного программирования

2. Расширения существующих языков программирования

mpC *(ИСП РАН): язык программирования, основанный на языке C, предоставляющий средства создания параллельных программ для компьютеров с распределенной памятью. Ориентирован на неоднородные системы. Пользователь может задать топологию сети, распределение данных и вычислений и необходимые пересылки данных. Посылка сообщений организована с использованием интерфейса MPI. Информация, извлечённая из описания параллельного алгоритма, вместе с данными о реальной производительности процессоров и коммуникационных каналов, помогает системе программирования mpC найти эффективный способ отображения процессов mpC-программы на компьютеры сети.*

Технологии параллельного программирования

2. Расширения существующих языков программирования

*Языки **Fortran-DVM** и **C-DVM** (ИПМ РАН).*

DVM-система состоит из 5 основных компонентов: компиляторы, система поддержки выполнения параллельных программ, отладчик параллельных программ (сравнение промежуточных результатов), анализатор производительности, предсказатель производительности (предиктор).

- *высокоуровневая модель программирования;*
- *спецкомментарии, один вариант программы для параллельного и последовательного выполнения;*
- *основная работа по распределению вычислений и данных выполняется динамически системой поддержки;*
- *DVM-препроцессор переводит программу в Си (Фортран), расширенный функциями системы поддержки;*
- *для организации межпроцессного взаимодействия может использоваться одна из коммуникационных технологий (MPI, PVM, Router), что обеспечивает хорошую переносимость;*
- *директивы DVM: распределение данных, распределение вычислений, спецификация удалённых данных, редукционные операции.*

Технологии параллельного программирования

2. Расширения существующих языков программирования

CAF (Co-Array Fortran):

- *небольшое расширение языка Fortran, достаточное для разработки эффективных параллельных программ;*
- *модель SPMD;*
- *понятие co-array (распределённого массива), компоненты которого распределены по процессам; доступ к распределённым компонентам осуществляется по правилам работы с обычными массивами.*

UPC (Unified Parallel C):

- *расширение языка программирования Си;*
- *модель SPMD;*
- *модель PGAS (Partitioned Global Address Space, глобальное разделённое адресное пространство) - адресуемая глобальная память в виде логических разделов, причем каждый из разделов локален для каждого из процессоров;*
- *синхронизация и обеспечение консистентности памяти.*
- *https://upc-lang.org/*

Технологии параллельного программирования

2. Расширения существующих языков программирования

CUDA *(https://developer.nvidia.com) и* ***OpenCL*** *(http://opencl.org/) - расширения Си-подобных языков для программирования графических ускорителей.*

Linda *- параллельный язык программирования. Программа рассматривается как совокупность процессов, которые могут обмениваться данным через пространство кортежей. Используется совместно с другими языками высокого уровня как средство общения параллельных процессов.*

Технологии параллельного программирования

3. Специальные языки программирования

Occam *- язык параллельного программирования, ориентированный в первую очередь на написание программ для транспьютерных систем. Позволяет описать любую ЯПФ с помощью инструкций par, seq, отмечающих участки параллельных и последовательных процессов.*

НОРМА *(ИПМ РАН):*

- *декларативный язык, предназначенный для описания решения вычислительных задач сеточными методами;*
- *описание задач в нотации, близкой к исходной постановке проблемы математиком;*
- *не содержит традиционные конструкции языков программирования, фиксирующие порядок вычисления и/или иным образом «скрывающие/ограничивающие» параллелизм;*
- *язык с однократным присваиванием, нет конструкций вида X=X+1;*
- *распараллеливанием занимается компилятор, результат получается на Фортран+MPI, Фортран+PVM и др.*

Технологии параллельного программирования

3. Специальные языки программирования

Colamo *(НИИ МВС ЮФУ) - язык структурно-процедурного программирования высокого уровня реконфигурируемых вычислительных систем (на основе FPGA).*

Другие языки параллельного программирования: Chapel, X10, Fortress…

Технологии параллельного программирования

4. Библиотеки и интерфейсы, поддерживающие взаимодействие параллельных процессов

MPI (Message Passing Interface) *- хорошо стандартизованный механизм для построения программ по модели обмена сообщениями. Существуют стандартные «привязки» MPI к языкам C и Fortran. Существуют бесплатные и коммерческие реализации почти для всех суперкомпьютерных платформ, а также для сетей рабочих станций UNIX и Windows. В настоящее время MPI - наиболее широко используемый и динамично развивающийся интерфейс из своего класса.*

PVM (Parallel Virtual Machine) *- общедоступная библиотека, предоставляющая возможности управления процессами с помощью механизма передачи сообщений.*

Shmem *реализует схему работы над общей памятью с помощью операций Put/Get. Shmem включена в стандарт MPI 2.0 в качестве раздела «односторонние коммуникации».*

Технологии параллельного программирования

5. Средства и технологии для поддержки метакомпьютерных и распределённых вычислений

Globus, gLite, UNICORE, Condor, BOINC, X-Com, Map/Reduce…

Технологии параллельного программирования

6. Параллельные предметные библиотеки

PBLAS *- параллельные версии базовых процедур линейной алгебры (BLAS), уровней 1, 2, 3. Библиотека разработана в рамках проекта ScaLAPACK.*

ScaLAPACK *включает подмножество процедур LAPACK, переработанных для использования на параллельных компьютерах, включая: решение систем линейных уравнений, обращение матриц, ортогональные преобразования, поиск собственных значений и др.*

ATLAS - *оптимизированная библиотека, включающая реализацию процедур BLAS и часть функциональности LAPACK.*

MKL *- оптимизированная реализация BLAS от Intel.*

FFTW, DFFTPack *- быстрое преобразование Фурье.*

PETSc *- набор процедур и структур данных для параллельного решения научных задач с моделями, описываемыми в виде дифференциальных уравнений с частными производными.*

…

Технологии параллельного программирования

7. Инструментальные системы разработки

- *Интегрированные среды прототипирования, разработки и отладки параллельных программ (**CODE, Converse, DEEP, EDPEPPS, GRADE, HeNCE, Reactor, TRAPPER** и др.)*

- *Средства распознавания параллелизма в алгоритмах, средства автоматического и полуавтоматического распараллеливания последовательных программ (**BERT 77, FORGE, KAP, PIPS, VAST, V-Ray, AlgoView** и др.)*

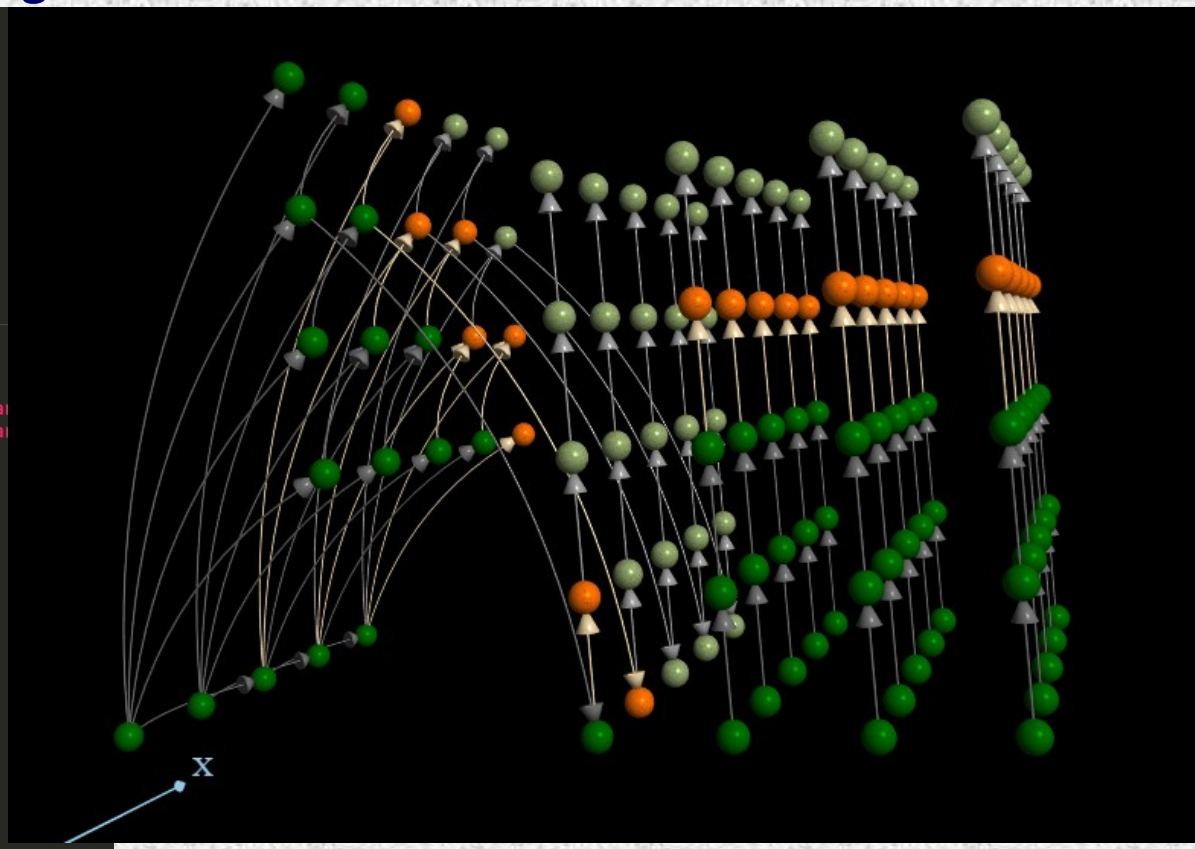
Технологии параллельного программирования

7. Инструментальные системы разработки

• *Система визуализации и анализа тонкой информационной структуры алгоритмов* ***AlgoView***

```
for(i = 2; i <= n+1; ++i)
    C[i] = C[i-2] * e;
for(i = 2; i <= n+1; ++i)
    for(j = 2; j <= m+1; ++j)
       B[i][j] = B[i-1][j-1] + C[i];
for(i = 2; i <= n+1; ++i){
    A[i][1][1]=B[i][m];
    for(j = 2; j <= m+1; ++j)
       for(k = 1; k <= n-1; ++k)
          A[i][j][k] = A[i][j][k] + A[i][j-1][k];
}
```

```
<?xml version = "1.0" encoding = "UTF-8"?>
<algo>
    <params>
        <param name = "N" type = "int" value = "5"></pa
        <param name = "M" type = "int" value = "4"></pa
    </params>
    <block id = "0" dims = "1">
        <arg name = "i" val = "2..N+1"></arg>
        <vertex condition = "" type = "1">
            <in src = "i - 2"></in>
        </vertex>
    </block>
    <block id = "1" dims = "2">
        <arg name = "i" val = "2..N+1"></arg>
        <arg name = "j" val = "2..M+1"></arg>
        <vertex condition = "" type = "1">
            <in src = "i - 1, j - 1"></in>
            <in bsrc = "0" src = "i"></in>
        </vertex>
    </block>
    <block id = "2" dims = "3">
        <arg name = "i" val = "2..N+1"></arg>
        <arg name = "j" val = "1..M+1"></arg>
        <arg name = "k" val = "1..N-1"></arg>
        <vertex condition = "(j == 1) and (k == 1)" type = "1">
            <in bsrc = "1" src = "i, M"></in>
        </vertex>
        <vertex condition = "(j > 1) or (k > 1)" type = "1">
            <in src = "i, j - 1, k"></in>
        </vertex>
    </block>
</algo>
```



•*Описание информационной структуры фрагмента на языке Algolang*

•*Интерактивная визуализация и анализ*

Технологии параллельного программирования

8. Специализированные пакеты и программные комплексы

FlowVision, ANSYS *для инженерный расчётов****;***
GAMESS, GAUSSIAN, PRIRODA, GROMACS, CHARMM *для квантово-химических расчётов;*
OpenFOAM *для вычислительной гидродинамики;*
…

Московский государственный университет имени М.В.Ломоносова
Факультет Вычислительной математики и кибернетики

СУПЕРКОМПЬЮТЕРЫ И ПАРАЛЛЕЛЬНАЯ ОБРАБОТКА ДАННЫХ (лекция 8)

А.С.Антонов

Вед. н.с. НИВЦ МГУ, к.ф.-м.н.

asa@parallel.ru

ВМК МГУ, 2024