Высокопроизводительные вычисления

- выбор вычислительной модели;
- использование эффективного вычислительного алгоритма;
- использование оптимальных структур данных и средств кодирования;
- использование оптимизированных библиотек;
- оптимизация при компиляции программы;
- оптимизация готовой программы на основе её выполнения;
- параллельное программирование;

Внедрение параллельного программирования

- Применение компиляторов, автоматически распараллеливающих фрагменты кода
- Применение специализированных библиотек, реализующих параллельные алгоритмы вычислений
- Применение специализированных пакетов расчета
- Применение технологий параллельного программирования.

Автоматически распараллеливающие компиляторы

- Некоторые компиляторы позволяют выполнять автоматическое распараллеливание фрагментов кода программы.
- Большинство распараллеливающих компиляторов работают со специализированными языками, которые упрощают задачу выделения независимых участков кода.
- Компиляторы универсальных языков высокого уровня (C/C++) позволяют распараллеливать только участки с явной независимой обработкой (например, циклическая обработка).

```
for (int i = 0; i < N; i++)
    a[i] = b[i] + c[i];
```

```
for (int i = 0; i < N; i++)
    x[a[i]] = x[b[i]] + x[c[i]];
```

Библиотеки

- BLAS и Lapack – библиотеки, реализующие базовые операции линейной алгебры (перемножение матриц, умножение матрицы на вектор и т.д.);
- ScaLapack – подмножество процедур библиотеки LAPACK, переработанных для использования на MPP-компьютерах, включая: решение систем линейных уравнений, обращение матриц, ортогональные преобразования, поиск собственных значений и др.
- 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.
- GALOPPS (Genetic ALgorithm Optimized for Portability and Parallelism System) - библиотека "обобщенных" генетических алгоритмов. Доступна многопоточная версия.
- NAMD - параллельная объектно-ориентированная библиотека для расчетов в области молекулярной динамики. Реализована на C++ с использованием шаблонов, а также Charm++ и Converse. Распространяется бесплатно. Доступны исходные тексты, а также откомпилированные двоичные файлы для платформ HP-UX, SGI Origin2000, Linux/x86, Solaris/SPARC, Cray T3E.

Пакеты расчета

- При решении типовых физико-технических задач в области гидродинамики, аэродинамики, молекулярной химии можно воспользоваться специализированными пакетами расчета. Современные пакеты учитывают возможность выполнения на параллельных архитектурах и применяют оптимизированные параллельные алгоритмы. Известными пакетами являются: ANSYS, Material Studio, OpenFOAM, GAMESS.
- Задачи инженерного анализа, прочности, теплофизики, деформации, упругости, пластичности, электромагнетизма (**ANSYS, LS-DYNA**)
- Задачи аэро- и гидродинамики, механики жидкостей и газов, горения и детонации (**FLUENT, FLOWVISION, CFX**)
- Задачи акустического анализа (**LMS Virtual Lab**)

Технологии параллельного программирования

Распределенные системы

- Стандарт MPI
- Erlang
- Go

Многопроцессорные системы

- Стандарт OpenMP
- Task Parallel Library (C#)
- Intel Cilk Plus
- Intel TBB

Гибридные системы (GPU)

- Nvidia CUDA
- OpenACC
- OpenCL
- MS AMP C++
- MC#

Технологии ПП для многоядерных систем

- Низкоуровневые средства (потоки)
  - Библиотеки для C/C++: pthreads, Windows Threads, boost::thread, std::thread (C++ 11);
  - Библиотеки для C#, Java, Python, ..
- Высокоуровневые средства (задачи)
  - Стандарт OpenMP (C++, Fortran)
  - Расширение Intel Cilk Plus (C++)
  - Библиотека Threading Building Blocks (C++)
  - Библиотека Parallel Primitives Library (C++)
  - Библиотека Task Parallel Library (C#)
  - ..

Потоки vs. задачи

- Работа с потоками предполагает:
  - декомпозиция задачи на части;
  - создание и управление потоками;
  - синхронизация потоков;
  - балансировка нагрузки потоков
  - агрегирование результатов;

- Работа с задачами упрощает разработку за счет **планировщика**, который самостоятельно подбирает оптимальное число потоков, выполняет декомпозицию и агрегирование данных, динамически балансирует нагрузку

Стандарт OpenMP

- Стандарт для многопроцессорного программирования на языках C/C++/Fortran
- Развивается с 1997 г. Последнее обновление: версия 4.5 (2015 г.)
- Поддерживается большинством компиляторов (Visual Studio, Intel C++, gcc, ..)
- Средства распараллеливания – директивы

```
// Параллельный цикл с OpenMP
    #pragma omp parallel for
    for (int i = 0; i < n; i++)
        y[i] = a * x[i] + y[i];
```

Intel Cilk Plus

- Расширение для языка C++ от Intel
- Средства распараллеливания: 3 ключевых слова, расширенная векторная нотация, гиперобъекты,..

```
// Параллельный цикл
cilk_for (int i = 0; i < n; i++)
        y[i] = a * x[i] + y[i];

// Расширенная индексная нотация:
 y[0:n] = a * x[0:n] + y[0:n];
```

Библиотека Intel TBB

- Кроссплатформенная библиотека шаблонов C++, разработанная компанией Intel для параллельного программирования.
- Включает типовые шаблоны распараллеливания, структуры данных для многопоточных сценариев, средства синхронизации

```
tbb::parallel_for(
    tbb::blocked_range<int>(0, n),
    [&](tbb::blocked_range<int> r) {
      for (int i=r.begin(); i != r.end(); ++i)
         y[i] = a * x[i] + y[i];
    }
);
```

Технологии программирования распределенных систем

- Интерфейсы передачи сообщений (MPI, WCF, ..)
- Библиотеки-оболочки над интерфейсом MPI (OO-MPI, boost mpi, MPI.NET, Global Arrays, ..)
- Языки с встроенной поддержкой обмена сообщениями (Go, Erlang, Chapel, X10, ..)

- Технологии MapReduce (apache hadoop, microsoft hdinsight, ..)

MPI-программа

```
int rank;
float msg = 0.0;
MPI_Status status;
MPI_Init();
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
if (rank == 0)
{
   msg = 3.14;
   printf("Sending message..\n", size);
   MPI_Send(&msg, 1, MPI_FLOAT, 1, 0, MPI_COMM_WORLD);
}
else if (rank == 1)
{
   MPI_Recv(&msg, 1, MPI_FLOAT, 0, 0,
                        MPI_COMM_WORLD, &status);
   printf("Received message: %f\n", msg);
}
MPI_Finalize();
```

MPI.NET

```
static void Main(string[] args)
  {
     using (new MPI.Environment(ref args))
     {
        Intracommunicator comm = Communicator.world;
        string name = MPI.Environment.ProcessorName;
        string[] names = comm.Gather(name, 0);
        if (comm.Rank == 0)
        {
           Array.Sort(names);
           foreach(string host in names)
              Console.WriteLine(host);
        }
     }
  }
```