Supercomputing Education: a Life Story about Technology

Prof. Vladimir Voevodin
Deputy Director RCC MSU
Head of Department on Supercomputers and Quantum Informatics CMC MSU
voevodin@parallel.ru
Parallel Processing
Pipelined and Parallel Processing
Synchronicity and Asynchrony
Various Parallel Computer Architectures

SMP

MPP
Variety of Multithreading, SIMD/MIMD
Barrier Synchronization
Shared and Distributed Memory
Student dream – to be able to do everything at the same time
Parallelism: Theory and Practice
Need for Super Performance
Simple but Fundamental Questions Behind Parallel Computing

How to solve problems faster?

How to solve problems *much faster*?

How to organize the work of a team of several people/cores/CPUs...?

Commercial company and Amdahl’s law...
What does it mean **reasonable sustained performance**?
• This level of performance is enough to solve the problem.

What does it mean **to keep sustained performance reasonably high**?
• You understand reasons why you achieve this level of performance,
  • You know reasons why you can't make the performance much higher.
We must describe everything that needs to be controlled.

Guarantee of coincidence between expectations and reality.

Four Key Components of Supercomputing Centers
(MSU Supercomputing Center)

DiMMon  OctoTron

ANALYTICS

OctoShell

We must control everything what is necessary to control efficiency permanently.
Analytics: Application Efficiency and Network Locality

Two similar runs:

- All input parameters are the same.
- Run time and MPI data transfers intensity are different.
- **Root cause:** poor network locality (6 switches instead of 3)
### Analytics: Abnormal Application Behavior

<table>
<thead>
<tr>
<th>Общая информация</th>
<th>Производительность</th>
<th>Базовые свойства</th>
</tr>
</thead>
<tbody>
<tr>
<td>Суперкомпьютер:</td>
<td><strong>lomonosov-2</strong></td>
<td><strong>метрика</strong></td>
</tr>
<tr>
<td>ID задачи:</td>
<td><strong>CENSORED</strong></td>
<td><strong>значение</strong></td>
</tr>
<tr>
<td>Логин:</td>
<td><strong>CENSORED</strong></td>
<td><strong>общая оценка</strong></td>
</tr>
<tr>
<td>Статус завершения задачи:</td>
<td>COMPLETED</td>
<td><strong>средняя загрузка ЦПУ (%)</strong></td>
</tr>
<tr>
<td>Раздел суперкомпьютера:</td>
<td>compute</td>
<td><strong>среднее LoadAVG</strong></td>
</tr>
<tr>
<td>Число ядер:</td>
<td>336</td>
<td><strong>среднее IPC</strong></td>
</tr>
<tr>
<td>Число узлов:</td>
<td>24</td>
<td><strong>средняя загрузка GPU (%)</strong></td>
</tr>
<tr>
<td>Постановка в очередь:</td>
<td>10/11/18 18:02:15</td>
<td><strong>интенсивность передачи данных по MPI (МБ/c)</strong></td>
</tr>
<tr>
<td>Начало счета:</td>
<td>10/11/18 19:35:17</td>
<td><strong>интенсивность чтений из файловой системы (МБ/с)</strong></td>
</tr>
<tr>
<td>Конец счета:</td>
<td>10/12/18 05:20:15</td>
<td><strong>интенсивность записи в файловую систему (МБ/с)</strong></td>
</tr>
<tr>
<td>Время счета (часы):</td>
<td>9.7</td>
<td></td>
</tr>
</tbody>
</table>

### Найденные потенциальные проблемы с эффективностью

В данном разделе приведен список проблем с эффективностью, которые были найдены для данной задачи. Для каждой проблемы приведено ее описание (какой признак возникновения проблемы был обнаружен), предположение (в чем, на наш взгляд, может заключаться причина возникновения проблемы) и рекомендация (что мы советуем сделать для ее устранения). Во многих случаях в рекомендации указано, какой тип дальнейшего анализа стоит проводить, реализацию этого функционала планируется выполнить в будущем.

Возможность оценки корректности и/или изменения обнаруженных проблем с эффективностью также планируется добавить в ближайшем будущем. Сейчас эта возможность реализована только на общей странице со списком задач.

<table>
<thead>
<tr>
<th>Тип</th>
<th>Описание</th>
<th>Предположение</th>
<th>Рекомендация</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Задача запущена в разделе для GPU задач, однако практически не использует графические процессоры.</td>
<td>Неправильно выбран раздел для задачи.</td>
<td>Рекомендуется сменить раздел.</td>
</tr>
<tr>
<td></td>
<td>Задача выполняется аномально неэффективно</td>
<td>Задача работает накорректно или зависла.</td>
<td>Рекомендуется проверить корректность запуска и при необходимости отменить его.</td>
</tr>
</tbody>
</table>

**КРАЙНЕ МНОГО!**
We must describe everything that needs to be controlled.

Guarantee of coincidence between expectations and reality.

We must control everything what is necessary to control efficiency permanently.

Life Conclusion:
All supercomputing centers should follow this structure otherwise considerable SC-resources will be wasted.
Supercomputing Modeling and technologies (summary on the course)

Brief contents:
• overview of problems that require supercomputer modeling;
• specific features of modern microprocessor architecture, new hardware technologies used in high-performance computing systems;
• introduction to parallel algorithm structure analysis;
• scalable algorithms for highly parallel computers: problems and perspectives;
• algorithmic approaches for supercomputer modeling in different areas: turbulent flows, molecular dynamics, climate changes, drug design…;
• additional chapters of MPI, OpenMP and CUDA parallel programming technologies for supercomputer modeling, and others…
• Lectures + practical assignments.

• One-semester course, 2 lectures per week.
• Lecturers are from various departments of CMC MSU and IT companies.

• Existing experience: three full cycles of the course in 2016 (246 students), 2017 (207 students) and 2018 (212).
Practical Assignments
(General comments)

- The goal of the assignments is to study and describe the parallel structure and properties of algorithms.

- Students are allowed to work in pairs.

- Variety of HPC platforms in use:
  - Supercomputers Lomonosov-2, Lomonosov,
  - Blue Gene/P,
  - Power8/Phi(KNL)/GPU(K40,Pascal)-based systems.

- Verification form of the assignments: not just to grade but to teach a proper approach to analyze properties of algorithms.
  (This requires tutors to have a much higher level of qualification and to dedicate more time to the assessment)
Assignment: **Describe the structure and properties of the chosen algorithm.**

What does it mean to describe an algorithm’s structure and properties? It is not a simple question at all...

Methodological basis of the assignment is the AlgoWiki Open Encyclopedia of Algorithm Properties (algowiki-project.org/en)
Open Encyclopedia of Parallel Algorithmic Features

Welcome! Join us!

AlgoWiki is an open encyclopedia of algorithms' properties and features of their implementations on different hardware and software platforms from mobile to extreme scale, which allows for collaboration with the worldwide computing community on algorithm descriptions.

AlgoWiki provides an exhaustive description of an algorithm. In addition to classical algorithm properties such as serial complexity, AlgoWiki also presents additional information, which together provides a complete description of the algorithm: its parallel complexity, parallel structure, determinacy, data locality, performance and scalability estimates, communication profiles for specific implementations, and many others.

Read more: About project.

Project structure

Algorithm classification — the main section of AlgoWiki which contains descriptions of all algorithms. Algorithms are added to the appropriate category of the classification, and classification is expanded with new sections if necessary.

Featured article

Cholesky decomposition

1 Properties and structure of the algorithm

1.1 General description

The Cholesky decomposition algorithm was first proposed by Andre-Louis Cholesky (October 15, 1875 - August 31, 1918) at the end of the First World War shortly before he was killed in battle. He was a French military officer and mathematician. The idea of this algorithm was published in 1924 by his fellow officer and, later, was used by Banachiewicz in 1938 [7]. In the Russian mathematical literature, the Cholesky decomposition is also known as the square-root method [1-3] due to the square root operations used in this decomposition and not used in Gaussian elimination.

Originally, the Cholesky decomposition was used only for dense real symmetric positive definite matrices. At present, the application of this decomposition is much wider. For example, it can also be employed for the case of Hermitian matrices. In order to increase the computing performance, its block versions are often applied.

In the case of sparse matrices, the Cholesky decomposition is also widely used as the main stage of a direct method for solving linear systems. In this case, the Cholesky decomposition provides a considerable saving of memory and computational resources.

Properties of the algorithm:
- Sequential complexity: $O(n^3)$
- Height of the parallel form: $O(n)$
- Width of the parallel form: $O(n^2)$
- Amount of input data: $\frac{n(n+1)}{2}$
- Amount of output data: $\frac{n(n+1)}{2}$

Guides to writing sections of the algorithm's description:
- Glossary
- Help with editing

Readiness of articles:
- Finished articles:
  - Single-qubit transform of a state vector
  - Two-sided Thomas algorithm, pointwise version
  - Poisson equation, solving with DFT
  - Thomas algorithm, pointwise version
  - Backward substitution

http://AlgoWiki-Project.org/en
Description of Algorithm Properties and Structure
(the assignment and AlgoWiki)

Contents [hide]

1 Properties and structure of the algorithm
   1.1 General description of the algorithm
   1.2 Mathematical description of the algorithm
   1.3 Computational kernel of the algorithm
   1.4 Macro structure of the algorithm
   1.5 Implementation scheme of the serial algorithm
   1.6 Serial complexity of the algorithm
   1.7 Information graph
   1.8 Parallelization resource of the algorithm
   1.9 Input and output data of the algorithm
   1.10 Properties of the algorithm

2 Software implementation of the algorithm
   2.1 Implementation peculiarities of the serial algorithm
   2.2 Locality of data and computations
   2.3 Possible methods and considerations for parallel implementation of the algorithm
   2.4 Scalability of the algorithm and its implementations
   2.5 Dynamic characteristics and efficiency of the algorithm implementation
   2.6 Conclusions for different classes of computer architecture
   2.7 Existing implementations of the algorithm

3 References
**Description of algorithms**

*(What properties of algorithms should be included in the description?)*

For positive definite Hermitian matrices *(symmetric matrices in the real case)*, we use the decomposition $A = LL^*$, where $L$ is the lower triangular matrix on $A = U^*U$, where $U$ is the upper triangular matrix. These forms of the Cholesky decomposition are equivalent in the sense of the amount of arithmetic operations and are different in the sense of data representation.

The essence of this decomposition consists in the implementation of formulas obtained uniquely for the elements of the matrix $L$ from features.

**General Description**

- Input data: a symmetric positive definite matrix $A$ whose elements are denoted by $a_{ij}$.
- Output data: the lower triangular matrix $L$ whose elements are denoted by $l_{ij}$.

The Cholesky algorithm can be represented in the form:

$$
\begin{align*}
    l_{11} &= \sqrt{a_{11}}, \\
    l_{1j} &= \frac{a_{1j}}{l_{11}}, \quad j \in [2, n], \\
    l_{ii} &= \sqrt{a_{ii} - \sum_{p=1}^{i-1} l_{ip}^2}, \quad i \in [2, n], \\
    l_{ij} &= \left( a_{ij} - \sum_{p=1}^{i-1} l_{ip} l_{jp} \right)/l_{ii}, \quad i \in [2, n-1], j \in [i+1, n].
\end{align*}
$$

The following number of operations should be performed for a matrix of order $n$ using a serial version of the Cholesky algorithm:

- $n$ square roots,
- $\frac{n(n-1)}{2}$ divisions,
- $\frac{n^3 - n}{6}$ multiplications and $\frac{n^3 - n}{6}$ additions (subtractions): the main amount of computational work.

A computational kernel of its serial version can be composed of $\frac{n(n-1)}{2}$ dot products.

**Mathematical Description**

**Serial Complexity**

**Computational Kernel**

**Performance**

**Information Structure**
Selection of 30 Algorithms for Description
(one algorithm for a student/pair)

• Jacobi’s method for the symmetric eigenvalue problem,
• Jacobi’s method for the singular value decomposition,
• the Lanczos algorithm with full reorthogonalization for the symmetric eigenvalue problem,
• Gram–Schmidt orthogonalization process,
• GMRES (Generalized Minimal RESidual method) as an iterative method for solving systems of linear algebraic equations,
• QR algorithm for solving the algebraic eigenvalue problem,
• Newton’s method for solving systems of nonlinear equations,
• fast discrete Fourier transform,
• minimum spanning tree based clustering algorithm,
• k-means clustering algorithm,
and other algorithms...

(!) Each algorithm is accompanied by references to well-known sources that explain the algorithm:
- students have a reliable source of information,
- students and tutors clearly understand what specific algorithm should be described.
Assignment: Studying the scalability of algorithms and their implementations on various computing platforms when changing problem size and a number of processors available.

Students needed to perform a series of computational experiments, collect the relevant data, plot graphs, interpret them correctly, and draw conclusions on scalability of algorithms.
Practical Assignment in 2017
(general comments)

Problems. Subject of study are algorithms for graph problems:

• Single Source Shortest Path,
• Breadth-First Search,
• Page Rank,
• Minimum Spanning Tree,
• Strongly Connected Components.

Algorithms. Various algorithms for each problems, for example, algorithms for the “Single Source Shortest Path” problem:

• Bellman-Ford, Dijkstra’s, and Delta-Stepping.

Implementations. Up to 5 different ready-to-use implementations were offered for each algorithm.

Computer Platforms. SMP-nodes of all HPC platforms available at the MSU supercomputing center.

Input data sets. Each student had to conduct experiments with two types of graphs: RMAT and SSCA2 (parallel generators of graphs were provided).
Practical Assignment in 2017  
(general comments)

The final assignment for students:
for each of the two graph types RMAT and SSCA2, and the chosen combination:

Graph Problem  Algorithm  Implementation  Computer platform

it is required
• to build a chart showing the performance behavior (TEPS) in dependence on the number of processors (or threads) and the graph size;
• to find the combination of processor number and graph size that maximizes performance.

A special focus is on large graphs ($2^{25} - 2^{28}$ nodes):
- to avoid cache effects,
- highly important in various applications.
Practical Assignment in 2017 (graph problems, max performance)

Graph Problems | Methods | Algorithms | Implementations | Computers
--- | --- | --- | --- | ---
Breadth-first search | BFS | RCC for GPU | Lomonosov-2 (NVIDIA P100) | 11061 | RMAT | 2^22
Breadth-first search | BFS | GAP | Lomonosov-2 (NVIDIA K40) | 5350 | RMAT | 2^22
Breadth-first search | BFS | Ligra | Lomonosov-2 (NVIDIA K40) | 4168 | RMAT | 2^22
Minimum Spanning Tree | Boruvka's | RCC for GPU | Lomonosov-2 (NVIDIA P100) | 3793 | RMAT | 2^26
Single Source Shortest Path | Bellman–Ford | RCC for GPU | Lomonosov (NVIDIA 2090) | 1309,0 | RMAT | 2^26
Strongly Connected Components | Shiloach–Vishkin | Ligra | Lomonosov-2 (x86) | 1307,00 | RMAT | 2^26
Single Source Shortest Path | Bellman–Ford | Ligra | Lomonosov-2 (x86) | 1035,0 | RMAT | 2^21
Single Source Shortest Path | Delta-Stepping | PBGL MPI | Cluster / "Angara" net | 809,5 | RMAT | 2^21
Page Rank | Page Rank | Nvidia nvGraph | Lomonosov-2 (NVIDIA K40) | 753 | RMAT | 2^18
Breadth-first search | BFS | RCC for CPU | NEC SX-ACE | 715 | RMAT | 2^22
Strongly Connected Components | Forward-Backward | RCC for GPU | Lomonosov-2 (NVIDIA P100) | 620,00 | RMAT | 2^21
Single Source Shortest Path | Delta-Stepping | GAP | Lomonosov-2 (x86) | 616,0 | RMAT | 2^21
Strongly Connected Components | Forward-Backward | RCC for CPU | Lomonosov-2 (x86) | 564,00 | RMAT | 2^24
Strongly Connected Components | Shiloach–Vishkin | GAP | Lomonosov-2 (x86) | 547,00 | RMAT | 2^24
Minimum Spanning Tree | Boruvka's | RCC for GPU | Lomonosov (NVIDIA 2090) | 204,441 | RMAT | 2^22
Page Rank | Page Rank | GAP | Lomonosov-2 (NVIDIA K40) | 172 | RMAT | 2^22
Breadth-first search | BFS | PBGL MPI | IBM BlueGene/P | 167 | SSCA-2 | 2^22
Breadth-first search | BFS | PBGL MPI | Lomonosov (x86) | 155 | SSCA-2 | 2^22
Minimum Spanning Tree | Boruvka's | RCC for CPU | Intel Xeon Phi (KNL) | 25,16 | SSCA-2 | 2^22
Single Source Shortest Path | Dijkstra's | PBGL MPI | IBM BlueGene/P | 8,9 | SSCA-2 | 2^22

- Competition between students is very useful!
- "Why is the performance worse in my case?"
- They are approaching to supercomputing co-design ideas...
Assignment 1 (2016) is **challenging in many aspects**...

- What does it mean “To extract and describe the total resource of parallelism of an algorithm”?

- How to show possible parallel execution?

- What does it mean “To build the information structure of an algorithm”?

- What does it mean “To draw the information structure of an algorithm”? What level of details for operations should be used?
Assignment 2 (2017) requires substantial computing resources since parallel programs need to be run multiple times to obtain performance data:

- different graph sizes,
- different number of processors,
- two graph types: RMAT and SSCA2,
- for each point several experiments need to be conducted to find a maximum value otherwise the resulting chart will contain artifacts.
When studying scalability, it is important to draw the students’ attention to explain all of the features on the performance charts: peaks, inflection points, asymptotes, etc. Detailed analysis is not simple, but if peculiarities consistently repeat between runs, then there must be an explanation.
**Practical Assignment: Results in 2016 and 2017**

*summary*

<table>
<thead>
<tr>
<th>Year – 2016</th>
<th>Year – 2017</th>
</tr>
</thead>
<tbody>
<tr>
<td>• 145 groups (246 students)</td>
<td>• 143 groups (207 students)</td>
</tr>
<tr>
<td>• Excellent – 59 (41%)</td>
<td>• Excellent – 121 (85%)</td>
</tr>
<tr>
<td>• Good – 36 (25%)</td>
<td>• Good – 15 (10%)</td>
</tr>
<tr>
<td>• Satisfactory – 48 (33%)</td>
<td>• Satisfactory – 5 (3%)</td>
</tr>
<tr>
<td>• Unsatisfactory – 2 (1%)</td>
<td>• Unsatisfactory – 2 (1%)</td>
</tr>
<tr>
<td>• Average grade – 4.03</td>
<td>• Average grade – 4.78</td>
</tr>
</tbody>
</table>

the lowest grade – 2 (Unsatisfactory), 3 (Satisfactory), 4 (Good), 5 (Excellent) – the highest grade

*Will we continue these assignments in the future? Yes!*
Useful Questions to Think about…

- What is the background of students in terms, for example, statistics and computer architecture?

- How do they reason about the performances of the underlying system and how do they correlate the hardware architecture with the algorithm?
  - Do they have a methodology lecture about performance measurements/tuning?

- How do the students use/reserve the platform? Do they get dedicated nodes for their experiments?

- What about reproducibility?
  - How do you verify the correctness of the experimental data the students provide?

- This seems like a lot of work both for students and teachers. Have you an estimation of the time spent 1) by students and 2) by teachers?

- I see that the second assignment involves OpenMP and MPI so I am a bit lost about the students' background and how they managed to finish those assignments.

- Do you plan to make this course available online for other instructors to adopt? OR How would you like other universities to adopt this course?
Parallel Computing at Schools: Feeling of the Coming Future

- About 600-700 schoolchildren come to excursions to MSU Supercomputing Center each year.

- March, 25th: 32 teachers + 60 schoolchildren from 28 schools of Balashikha town (Moscow region).
Supercomputing Education: a Life Story about Technology

Prof. Vladimir Voevodin
Deputy Director RCC MSU
Head of Department on Supercomputers and Quantum Informatics CMC MSU
voevodin@parallel.ru