
4.1 Main performance characteristics.

Possibility to see differences between parallel and sequential parts of the program (that you run on the multiprocessor computer) helps you to predict time that you will need to run this program on the one-processor computer. This time is called **useful time**. Next, we can calculate the main characteristic of parallel execution – **efficiency coefficient** that is equal to the ratio of useful time and **total time** of processor’s using (product of maximum time on a single processor and **number of processors**). The difference between the whole time of processors using and useful time is **lost time**. If the programmer doesn’t like his program’s efficiency coefficient he needs to analyze the parts of lost time and the causes of appearance.

Parts of lost time:

- The losses because of the **insufficient parallelism** (duplicating calculations on several processors). Duplicating of calculations can be occurred in two cases. First, sequential parts of the program are executed by all processors. Second, iterations of some parallel loops can be duplicating (partly or completely) by programmer’s instruction.

- The losses because of the interprocessor’s exchanges (**communication**).

- The losses because of the idle of some processors (one processor can execute its part of calculations quicker than others) – **idle time**.

Time of interprocessor’s exchanges can consist not only of sending time, but also consists of time that is wasting, because the receive operation on one processor was executed before the send operation on the other one. Such a situation is called **mistiming** and has different causes.

Because the mistiming losses are the main part of all communications, it is very important to give programmer information to value these losses and find the causes. To value the size of these losses for every collective operation you should calculate the potential losses because of non-simultaneous execution – time that would be lost on synchronization by all processors if the execution of every collective operation started with synchronization. Time for sending synchronizing messages is ignored.

To summarize potential losses that can appear because of non-simultaneous execution of collective operations on different processors, we can use special characteristic – **synchronization**.
The main cause of mistiming is deregulation of processor’s loading. Deregulation can appear when calculations in parallel cycle were distributed unequally…

Main performance characteristics are integral ones that help you to value the size program’s parallelization and main reserves for its increasing. But complicated tasks sometimes need not only integral characteristics. In this case programmer wants to get more detailed information about program execution and its parts.

4.2 Methods of Performance Debugging.

For analyzing efficiency of complicated parallel programs you not only need the whole program characteristics, but also need to get these characteristics conformably to the separate parts of the program.

So, we create means that can represent programs execution in the hierarchy of intervals.

Intervals:
Execution of the whole program is assumed as the interval of the highest (zero) level. This interval may include few intervals of the next (first) level. It can be parallel or sequential cycle and, also, every marked by the programmer sequence of operators (execution of these operators always starts with the first and ends with the last one). All characteristics that were described above are calculated not only for the whole program, but for every interval.

User can create fragmentation on intervals during program’s compilation. He/she can also set such regimes when sequential cycles that include parallel ones are intervals, or all sequential cycles at all, or marked sequence of operations.

5. The Mean for Performance Analysis of MPI program.

5.1 Task Statement.

The main purpose of this work is creating the experimental system for performance debugging of MPI programs.

Input data is traces created by the DVM system for functional debugging of MPI-programs. In this traces we can find calls of MPI-functions and time of their work. To get all characteristics we need extra information about which calculations are sequential and which are parallel (duplicating on each processor). These instructions should be made in such a way that they didn’t disturb the right and efficient MPI-program execution. The intervals bounds should be made the same way.
How to create intervals.
For this we use the pair MPI_Send() and MPI_Recv() for begin and the same pair for end of the interval. Send/receive operations send/receive to it or for itself (I mean that the number of source/recipient processor is the same as the current number).

Moreover, the tag message looks like that:

\[ \text{TAG} = 0x(aa)(id)(aa/bb). \]

Tag is a 4-byte integer number. First byte – 0xaa. It helps to differ it from the other messages. The last byte should be 0xaa – the beginning of the interval or, 0xbb – end. In the middle of the tag, special id of the interval (2 bytes), you can use it for instance to mark iterations of the cycle.

We use such method because:
- It is always in the trace (I mean send/receive operations). Some special functions can be ignored by the tracer like MPI_Pcontrol() (in the current version).
- Only 100 tics per one usage.
- Simple and do not need any special means except of the standard MPI functions.

Analyzer’s steps.

There are few steps in the work of analyzer.

Step 1.

Traces from all processors should be processed and main characteristics should be calculated:

Main characteristics and components.

Parallelization efficiency is equal to the ratio of useful time and total time of processors using.

Execution time.

Number of Processors (Processors).

Total time - (Execution time) * (Processors).

Productive time – prediction for one-processor execution.

Lost time.

Communication and all its components.

Idle time.

Load_Imbalance.

Execution Characteristics on Each Processor.
**Lost time** – the sum of the losses because of the insufficient parallelism (*User insufficient_par*), and system losses (*Sys insufficient_par*), (**Communication**) and (**Idle time**).

**Idle time** – is the difference between the maximum time of interval execution (on some processor) and time of execution on the given processor.

Total communication time (**Communication**).

**Real synchronization** – real synchronization losses.

Potential losses because of the time differences (**Variation**).

**Load imbalance** – calculated as the difference between the maximum time (**CPU+MPI**) and corresponding sum on the given processor.

Interval **Execution time**.

Useful processor time (**User CPU_time**).

Useful system time (**MPI time**).

Number of processors that was used for this interval (**Processors**).

Time of communication for all types of collective operations.

---

Step 2

The results should be given in text form. Such form is the best way for the first analysis of the parallel program execution.

---

Step 3

Visualization for performance analysis.

Visualization would give user the possibility to see the graphical form of the combined characteristics and help to research them – to see history of execution (detailed or not) and understand how they were calculated.

---

**Example of the result.**

Text information in the output file.

---

Interval (LineNumber = 153 SourceFile = exch.c) Level=0 EXE_Count=1

---Main Characteristics---

Parallelization Efficiency 0.978833
Execution Time 2.079975
Processors 4
Total Time 8.319900
Productive Time 8.143794 (**CPU** **MPI**)

MPI on one processor is productive, on the others – lost time.

Lost Time 0.176106
---MPI Time 0.173490
---Idle Time 0.002616
Communication Time 0.076563
****SendRecv Time 0.013295
****CollectiveAll Time 0.063268
****AllToAll Time 0.000000
Potential Sync. 0.068763
Time Variation 0.001790
Time of Start 0.000000

---Comparative Characteristics---

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Tmin</th>
<th>Nproc</th>
<th>Tmax</th>
<th>Nproc</th>
<th>Tmid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lost Time</td>
<td>0.033087</td>
<td>3</td>
<td>0.060057</td>
<td>0</td>
<td>0.044026</td>
</tr>
<tr>
<td>Idle Time</td>
<td>0.000000</td>
<td>1</td>
<td>0.000898</td>
<td>0</td>
<td>0.000654</td>
</tr>
<tr>
<td>Comm. Time</td>
<td>0.006597</td>
<td>3</td>
<td>0.034854</td>
<td>0</td>
<td>0.019140</td>
</tr>
<tr>
<td>MPI Time</td>
<td>0.032259</td>
<td>3</td>
<td>0.059159</td>
<td>0</td>
<td>0.043372</td>
</tr>
<tr>
<td>Potential Sync.</td>
<td>0.001800</td>
<td>0</td>
<td>0.029369</td>
<td>3</td>
<td>0.017190</td>
</tr>
<tr>
<td>Time variation</td>
<td>0.000161</td>
<td>1</td>
<td>0.000607</td>
<td>3</td>
<td>0.000447</td>
</tr>
<tr>
<td>Time of Start</td>
<td>0.000000</td>
<td>0</td>
<td>0.000000</td>
<td>0</td>
<td>0.000000</td>
</tr>
</tbody>
</table>

Visualization

Next, after all characteristics were collected we could use visualization. This step is needed in spite of the text file has already had all useful information. Because when the number of intervals is big, it’s hard to use text information. I have decided to show intervals in the tree form (like a file system). So, every interval can be getting from its parent one and is shown right to the upper level. Also, when you press on the interval you can get information about it. It makes easier to analyze execution.

Also, very useful for analysis is so called TimeLine (ordered by time list of events). All the actions have their own color on TimeLine and are shown according to the time when they were occurred. It gives you possibility not only to
get interesting region, but interesting action. Using Tool tips user can get information about every action (user’s code or MPI function) and the name of the function.

Pict.2. TimeLine

**Conclusion.**

In this work I’ve researched the possibilities of performance analysis of MPI-programs and created the tool – performance analyzer for MPI-programs based on the traces collected during program execution. The size of the result code = 6500 lines (C++ code).

The program was tested using the tests delivered with MPI realizations and the NAS (NPB2.3) tests with adding the margins for intervals (described above).

During this work I have:

- Analyzed most of the modern means for parallel performance analysis (advantages and disadvantages) Part1 and Part2.
- Created algorithms for analyzing and collecting useful characteristics. Part3 and Part4.
- Created the Tool with such abilities:
  1. Shows the program execution in the interval-tree form.
  2. Collects and shows collected characteristics for the chosen interval.
  3. All main information in the text form (in the file).
  4. Shows TimeLine.