POP Workflow

Alesja Dammer, IT-Center, RWTH Aachen
• A horizontal Center of Excellence
  • Transversal across application areas, platforms, scales

• Providing Performance Optimization and Productivity services
  • Precise understanding of application and system behavior
  • Suggestion on how to refactor code in the most productive way

• For (your?) academic AND industrial code(s) !
Services provided by the CoE

Application Performance Audit
- Primary service
- Identify performance issues of customer code (at customer site)
- Small effort (< 1 month)

Application Performance Plan
- Follow-up on the audit service
- Identifies the root causes of the issues found and qualifies and quantifies approaches to address them
- Longer effort (1-3 months)

Proof-of-Concept
- Experiments and mock-up tests for customer codes
- Kernel extraction, parallelization, mini-apps experiments to show effect of proposed optimizations
- 6 months effort

Training
Project Partners
Performance Audit: how we start

- Customer request
- Getting the code, data set, test setups
- Code porting
- Instrumentation/Profiling/Tracing

**Tools:**
- [https://tools.bsc.es](https://tools.bsc.es)
- [http://www.scalasca.org](http://www.scalasca.org)

- Application is ready for the analysis
Workflow of Performance Analysis

• First brief **overview** of program performance
• Understanding of **application structure** and finding the **region of interest**

• **Parallel Analysis**
  • Load Balance among processes
  • Efficiency Metrics
  • Communications

• **Serial Performance**
  • Instruction per Cycles (IPC) for main procedures
  • Memory behavior
  • Vectorization behavior
Brief overview of Program Performance

High level tools:

- **Intel Snapshot** (works only with Intel MPI)
- **ARM Performance Reports** (limited on number of licenses)

  - CPU: Scalar/Vector operations
  - MPI: time in MPI operations
  - OpenMP: computation / synchronization
  - Memory behavior: remote accesses, memory bound
  - CPI
  - FPU Utilization: FLOPs, Vector utilization
  - IO: read/write time
Tools:
- Scorep/Scalasca/Cube/Vampir
- EXTRAE/Paraver
- Intel VTune

Vampir-Trace

Cube
Parallel Analysis

• Applying **Scalasca** analysis on Scorep results or applying of a tool **Dimemas** on Extrae results

• **Global Efficiency (GE) = PE * CompE**
  • Parallel Efficiency (PE) = ComE * LB
    • Load Balance (LB)
    • Communication Efficiency (ComE) = SE * TE
      • Serialization Efficiency (SE)
      • Transfer Efficiency (TE)
  • Computational Efficiency (CompE)

• Values are from 0 to 1
  • The higher the better
  • Less than 0.8 -> a performance issue
    • Analyze MPI communications, waiting time, etc.
Load Balance

• Load Balance of
  • Computation time
  • Executed instructions

\[
LB = \frac{\text{Avg}}{\text{Max}} = \frac{\text{SUM}}{\#\text{Procs}} \frac{\text{Max}}{\text{Max}}
\]
**Efficiency Metrics**

- **Serialization Efficiency** reflects the loss of efficiency due to dependencies among processes.

\[
SE = \frac{\text{max comp time}}{\text{comp time of CP}}
\]

CP-critical path (communication time excluded) in blue.

Example:

\[
SE = \max\left\{ (\text{comp1} + \text{comp2} + \text{comp3}), \ (\text{comp4} + \text{comp5} + \text{comp6}) \right\}
\]

\[
\frac{\text{comp1} + \text{comp5} + \text{comp3}}{\text{comp1} + \text{comp5} + \text{comp3}}
\]
Efficiency Metrics

- **Transfer Efficiency** reflects the loss of efficiency due to actual data transfer

\[ TE = \frac{\text{comp time of CP}}{\text{total wall time}} \]

Example: \[ TE = \frac{\text{comp1} + \text{comp5} + \text{comp3}}{T} \]
Efficiency Metrics

- **Communication Efficiency** reflects the loss of efficiency by MPI communication and synchronization

\[
ComE = SE \times TE = \frac{\text{max comp time}}{\text{total wall time}}
\]

- **Parallel Efficiency**

\[
PE = LB \times ComE
\]

- **Global Efficiency** executing with \( p \) processes

\[
GE_p = \text{CompE}_p \times PE_p
\]
• **Computation Efficiency** reflects the loss of efficiency due to increasing the number of cores

\[
\text{CompE}_2 = \frac{\text{CompT}_1}{\text{CompT}_2}
\]

Example:
• Execution with 1 processes: \( \text{CompT}_1 = \text{comp1} + \text{comp2} \)

\[
\begin{align*}
\text{CompT}_1 &= \text{comp1} + \text{comp2} \\
\end{align*}
\]

• Execution with 2 processes: \( \text{CompT}_2 = \text{comp3} + \text{comp4} + \text{comp5} + \text{comp6} \)

\[
\begin{align*}
\text{CompE}_2 &= \frac{\text{comp1} + \text{comp2}}{\text{comp3} + \text{comp4} + \text{comp5} + \text{comp6}} \\
\end{align*}
\]
• Look at
  • trace time line
  • time sharing among MPI operations
Serial Performance

• Instructions per Cycle (IPC)

• using of PAPI counters in configuration file or as environment variable

• Under 1.0 → performance issue -> analyze cache behavior, memory bandwidth
Serial Performance

• L1/L2/L3 cache miss ratio
  = cache misses / cache accesses

• Use of memory bandwidth
  • Likwid

• Vectorization
  • Compiler vectorization report
  • Intel Advisor

<table>
<thead>
<tr>
<th>Loop metrics</th>
<th>Value</th>
<th>Efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total CPU time</td>
<td>3337.46s</td>
<td>100.0%</td>
</tr>
<tr>
<td>Time in 177 vectorized loops</td>
<td>375.56s</td>
<td>11.3%</td>
</tr>
<tr>
<td>Time in scalar code</td>
<td>2961.91s</td>
<td>88.7%</td>
</tr>
</tbody>
</table>

Vectorization Gain/Efficiency

Vectorized Loops Gain/Efficiency | 1.73x
Program Theoretical Gain | ~88%
Other Observations

- **OpenMP Performance**: Intel VTune, Extrae/Paraver, Scorep (limited)
  - Load Balance among threads
  - FLOPs and IPC
  - NUMA effects

- **IO behavior**: Vampir-Trace, Darshan
  - File operations by master process
  - Read/write sequentially or in parallel
Next Phases

• **Performance Plan**
  - Identifies the root causes of the issues found in PA
  - Qualifies and quantifies approaches to address the issues
  - Deeper analysis of found issues
  - Suggestions for implementation of possible solutions

• **Proof-of-Concept**
  - Implementation of solutions
  - Performance analysis of optimized application
  - Comparing metrics to first analysis results
POP Effort

• In 2,5 years were performed
  • Performance Audit for over 100 applications
  • 16 Performance Plans
  • 17 Proof-of-Concepts