Automatic performance modeling is back in town

Felix Wolf¹, Alexandru Calotoiu¹, Torsten Hoefler²
¹TU Darmstadt, ²ETH Zurich

Motivation

**Performance model** = formula that expresses relevant performance metrics as a function of one or more execution parameters

- Manual creation challenging
  - Identify kernels
    - Incomplete coverage
  - Create models
    - Laborious, difficult
**Historical background**

**Catwalk (2013-2017)**
- TU Darmstadt, ETH Zürich, GU Frankfurt
- Goal: automatic empirical performance modeling
- Main result: performance modeling tool **Extra-P**

**ExtraPeak (2017-2020)**
- TU Darmstadt, ETH Zürich
- Associated with SPPEXA since August 2017
- Goal: allow Extra-P to create models with **multiple parameters**

---

**Automatic empirical performance modeling**

**Small-scale measurements**

**Kernel**

[2 of 40]

**Model [s]**

<table>
<thead>
<tr>
<th>sweep → MPI_Recv</th>
<th>4.03√p</th>
</tr>
</thead>
<tbody>
<tr>
<td>sweep</td>
<td>582.19</td>
</tr>
</tbody>
</table>

**Performance model normal form (PMNF)**

\[ f(p) = \sum_{k=1}^{n} c_k \cdot p^k \cdot \log_{2}^k(p) \]
While we were away…

1. Performance models with multiple parameters
2. Automatic configuration of the search space
3. Segmented models
4. Iso-efficiency modeling

Models with more than one parameter

\[ f(x_1, \ldots, x_m) = \sum_{k=1}^{n} c_k \prod_{l=1}^{m} x_{l}^{j_{kl}} \cdot \log_2^{j_{kl}}(x_l) \]

Search space explosion

- Total number of hypotheses to search: 34,786,300,841,019
- Too slow for any practical purpose
Search space reduction through heuristics

- **Hierarchical search** – Assumes the best multi-parameter model is created out of the combination of the best single parameter hypothesis for each parameter.

- **Modified golden section search** – Speeds up the single parameter search by ordering the hypothesis space and then using a variant of binary search to find the model in logarithmic time rather than linear time.

---

Automatic configuration of the search space

**Simplified PMNF**

- Constant and “lead order” term
  \[ f(p) = c_0 + c_1 \cdot p^\alpha \cdot \log_2 p \]
- \( c_0 \) and \( c_1 \) are determined by regression
- \( \alpha \) and \( \beta \) found automatically via iterative refinement

**Results**

- 4453 models
- 49% remain unchanged
- 39% get better
- 12% get worse
- Improvements in every individual case study
Segmented models

Efficiency of task-based applications

const. efficiency = \frac{S}{p}
Efficiency of task-based applications

- Input size
- Task graph
- Core count

const. efficiency = \frac{S}{p}

Modeled efficiency functions

- \( E_{ub}(p,n) \) – upper bound based on avg. parallelism
- \( \Delta_{str} = E_{ub}(p,n) - E_{cf}(p,n) \) – structural discrepancy: optimization potential at the level of the task graph
- \( E_{cf}(p,n) \) – contention-free replays
- \( \Delta_{con} = E_{cf}(p,n) - E_{ac}(p,n) \) – contention discrepancy: severity of resource contention
- \( E_{ac}(p,n) \) – reflects actual performance
Iso-efficiency in co-design

<table>
<thead>
<tr>
<th>App.</th>
<th>Model</th>
<th>Input size for $p = 60$, $E = 0.8$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fibonacci</td>
<td>$E_{ac} = 0.98 - 5.11 \cdot 10^{-3} p^{0.25} + 1.76 \cdot 10^{-3} p^{0.25} \log n$</td>
<td>51</td>
</tr>
<tr>
<td></td>
<td>$E_{cf} = 0.97 - 1.46 \cdot 10^{-2} p^{0.25} + 9.26 \cdot 10^{-3} p^{0.25} \log n$</td>
<td>51</td>
</tr>
<tr>
<td></td>
<td>$E_{ub} = \min {1, 25.48 + 0.49 (0.25 n^{0.75} p^{-1/3}) }$</td>
<td>49</td>
</tr>
<tr>
<td>Strassen</td>
<td>$E_{ac} = 1.55 - 1.02 p^{0.25} + 4.59 \cdot 10^{-2} p^{0.25} \log n$</td>
<td>83,600 x 83,600</td>
</tr>
<tr>
<td></td>
<td>$E_{cf} = 1.26 - 0.65 p^{0.25} + 3.89 \cdot 10^{-2} p^{0.25} \log n$</td>
<td>12,680 x 12,680</td>
</tr>
<tr>
<td></td>
<td>$E_{ub} = \min {1, 0.25 n^{0.75} p^{-1} }$</td>
<td>1,200 x 1,200</td>
</tr>
</tbody>
</table>

For example (Strassen): $E_{ac} = 1.55 - 1.02 p^{0.25} + 4.59 \cdot 10^{-2} p^{0.25} \log n$

Let $E = 0.8$ and $p = 60$: $0.8 = 1.55 - 1.02 \cdot 60^{0.25} + 4.59 \cdot 10^{-2} \cdot 60^{0.25} \log n$

After solving: $n = 83,600$

Shudler et al. PPoPP’17

Extra-P 3.0

- GUI improvements, better stability, additional features
- Tutorials available through VI-HPS and upon request

http://www.scalasca.org/software/extra-p/download.html
Ongoing work

- Requirements engineering for co-design
  - Applications requirements scale differently for different resources (e.g., network, processor)
- Goal
  - Model portable requirement metrics
  - Extrapolate to hypothetical system
  - Pay attention to memory locality
- Cost-effective sampling of the performance space
  - Use machine learning to optimize cost or accuracy

Related projects

- TaLPas (BMBF, 2017-2019)
  - Extra-P used to optimize scheduling decisions
  - Partners: TU Munich, U Hamburg, TU Darmstadt, TU Kaiserslautern, U Paderborn, U Stuttgart
- EPE (DFG, 2016-2019)
  - Part of the DFG Program Performance Engineering for Scientific Software
  - Extra-P at the center of scalability service
  - Partners: TU Darmstadt, GU Frankfurt, JGU Mainz, TU Kaiserslautern
## Publications related to Extra-P

<table>
<thead>
<tr>
<th>Publication</th>
<th>Conference/Location</th>
</tr>
</thead>
</table>

4/11/2018 | Department of Computer Science | Laboratory for Parallel Programming | Felix Wolf | 17