Making Sense of Performance Numbers

Georg Hager
Erlangen Regional Computing Center (RRZE)
Friedrich-Alexander-Universität Erlangen-Nürnberg

OpenMPCon 2018
Barcelona, 2018-09-25
Agenda

- Performance Engineering
- Analytic performance models
- A simple example: Sparse matrix-vector multiplication
- Another example: Sparse matrix-transpose vector multiplication
- An advanced chip-level model: ECM
- Yet another example: Composite modeling of a (P)CG solver
Performance Engineering

- Performance Engineering (PE): a structured process based on analytic (white-/gray-box) models to optimize/parallelize codes

- Basic questions addressed by analytic performance models
  - What is the bottleneck? → optimization technique
  - What is the next bottleneck? → performance potential of the optimization
  - Am I done? What about other hardware?
  - Impact of processor frequency and socket scalability → Appropriate execution parameters, energy-optimized operating point

- Engineering for Performance in High Performance Computing (Bill Gropp; PASC 2015): https://www.youtube.com/watch?v=sadfSARXSC0
Motivation for analytic modeling

- Advantages of analytic models
  - Identification of universality
  - Identification of governing mechanisms
  - Insight via model nature
  - Insight via model failure

- Performance models
  - Microarchitecture analysis: Determine bottlenecks and influencing factors
  - Design space exploration: What would happen if resource $X$ were improved?
NODE LEVEL PERFORMANCE MODEL FOR SPARSE MATRIX (TRANSPOSE) VECTOR MULTIPLY
SpMV – Roofline Model (Comp. Intensity)

Sparse MVM in double precision with CRS data storage:
\( (N_{nzr} : \text{avg. non-zeros per row}) \)

\[
\text{do } i = 1, N_r \\
\quad \text{do } j = \text{row_ptr}(i), \text{row_ptr}(i+1) - 1 \\
\quad \quad C(i) = C(i) + \text{val}(j) \times \text{B}(:, \text{col_idx}(j)) \\
\quad \text{enddo} \\
\text{enddo}
\]

Double precision computational intensity

- \( \alpha \) quantifies traffic for loading RHS (\( \mathbf{B} \))
  - \( \alpha = 0 \) → RHS is in cache
  - \( \alpha = 1/N_{nzr} \) → RHS loaded once
  - \( \alpha = 1 \) → no cache
  - \( \alpha > 1 \) → Houston, we have a problem!
- “Expected” performance = \( b_S \times I_{\text{CRS}} \) (Roofline model)

\[
I^{DP}_{\text{CRS}} = \frac{2}{8 + 4 + 8\alpha + 20/N_{nzr}} \text{ flops/byte}
\]

See
SpMV – Quantifying RHS impact ($\alpha$)

\[ I_{CRS}^{DP} = \frac{2}{8 + 4 + 8\alpha + 20/N_{nzr}} \text{flops per byte} = \frac{N_{nz} \cdot 2 \text{flops}}{V_{meas}} \]

$V_{meas}$ is measured overall memory data traffic (using, e.g., likwid)

\[ \alpha = \frac{1}{4} \left( \frac{V_{meas}}{N_{nz} \cdot 2 \text{ bytes}} - 6 - \frac{10}{N_{nzr}} \right) \]

Example: kkt_power matrix from the UoF collection

- Intel Xeon Sandy Bridge (8c; 20 MB L3)
- $N_{nz} = 14.6 \cdot 10^6$, $N_{nzr} = 7.1$
- $V_{meas} \approx 258$ MB $\Rightarrow \alpha = 0.36$, $\alpha N_{nzr} = 2.5$
- $\Rightarrow$ RHS is loaded 2.5 times from memory

- but: $\frac{I_{CRS}^{DP}(1/N_{nzr})}{I_{CRS}^{DP}(\alpha)} = 1.11$

**11% extra traffic $\Rightarrow$ optimization potential!**
SpMV – understanding performance

Intel Xeon E5-2680 (8c@2.7 GHz; 20 MB L3)
Assumption: \( \alpha = \frac{1}{N_{nzr}} \rightarrow P = I_{CRS}^{DP} \times b \)

<table>
<thead>
<tr>
<th>#</th>
<th>Test case</th>
<th>(N)</th>
<th>(N_{nzr})</th>
<th>(N_{nzr})</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>RM07R</td>
<td>381,689</td>
<td>37,464,962</td>
<td>98.16</td>
</tr>
<tr>
<td>2</td>
<td>kkt_power</td>
<td>2,063,494</td>
<td>14,612,663</td>
<td>7.08</td>
</tr>
<tr>
<td>3</td>
<td>Hamrle3</td>
<td>1,447,360</td>
<td>5,514,242</td>
<td>3.81</td>
</tr>
<tr>
<td>4</td>
<td>ML_Geer</td>
<td>1,504,002</td>
<td>110,879,972</td>
<td>73.72</td>
</tr>
<tr>
<td>5</td>
<td>pwtk</td>
<td>217,918</td>
<td>11,634,424</td>
<td>53.39</td>
</tr>
<tr>
<td>6</td>
<td>shipsec1</td>
<td>140,874</td>
<td>7,813,404</td>
<td>55.46</td>
</tr>
<tr>
<td>7</td>
<td>consph</td>
<td>83,334</td>
<td>6,010,480</td>
<td>72.13</td>
</tr>
<tr>
<td>8</td>
<td>pdb1HYS</td>
<td>36,417</td>
<td>4,344,765</td>
<td>119.31</td>
</tr>
<tr>
<td>9</td>
<td>cunt</td>
<td>62,451</td>
<td>4,007,383</td>
<td>64.17</td>
</tr>
<tr>
<td>10</td>
<td>cop20k_A</td>
<td>121,192</td>
<td>2,624,331</td>
<td>21.65</td>
</tr>
<tr>
<td>11</td>
<td>rma10</td>
<td>46,835</td>
<td>2,374,001</td>
<td>50.69</td>
</tr>
<tr>
<td>12</td>
<td>mc2depi</td>
<td>525,825</td>
<td>2,100,225</td>
<td>3.99</td>
</tr>
<tr>
<td>13</td>
<td>qcd5.4</td>
<td>49,152</td>
<td>1,916,928</td>
<td>39.00</td>
</tr>
<tr>
<td>14</td>
<td>mac_econ_fwd500</td>
<td>206,500</td>
<td>1,273,389</td>
<td>6.17</td>
</tr>
<tr>
<td>15</td>
<td>scircuit</td>
<td>170,998</td>
<td>958,936</td>
<td>5.61</td>
</tr>
<tr>
<td>16</td>
<td>rail4284</td>
<td>4,284\times</td>
<td>11,279,748</td>
<td>2,632.99</td>
</tr>
<tr>
<td>17</td>
<td>dense2</td>
<td>2,000</td>
<td>4,000,000</td>
<td>2,000.00</td>
</tr>
<tr>
<td>18</td>
<td>webbase-1M</td>
<td>1,000,005</td>
<td>3,105,536</td>
<td>3.11</td>
</tr>
</tbody>
</table>

4 corner case matrices from UoF collection
Williams collection http://www.nvidia.com/content/NV_Research/matrices.zip

SpMV – understanding performance

Socket scaling – more recent architectures

PWTK matrix \( (N_{nzr} = 53 & \alpha = \frac{1}{N_{nzr}} \Rightarrow I_{CRS}^{DP} = \frac{2F}{12.5B} \Rightarrow P = I_{CRS}^{DP} \times b) \)

\[
b = 68 \text{ GB/s}
\]

Model covers saturation regime – full chip!
SpMTransposeV – understanding performance

\[
\text{do } i = 1,N_r \\
\text{ do } j = \text{row_ptr}(i), \text{row_ptr}(i+1) - 1 \\
\quad \text{C(colIdx}(j)) = \text{C(colIdx}(j)) + \text{val}(j) * B(i) \\
\text{ enddo} \\
\text{ enddo}
\]

\[
I^{DP}_{CRS} = \frac{2}{8 + 4 + 16\alpha + 12/N_{nzr}} \text{ byte}
\]

- Assume \(\alpha = \frac{1}{N_{nzr}}\)
- Does the model fail?
- Parallelization ?! (write conflicts!)

![Graph showing performance and Intel MKL speed with b=68 GB/s](image-url)
SpMTV – understanding performance

\[
\begin{align*}
  & \text{do } i = 1, N_r \\
  & \quad \text{do } j = \text{row_ptr}(i), \text{row_ptr}(i+1) - 1 \\
  & \quad \quad C(\text{col_idx}(j)) = C(\text{col_idx}(j)) + \text{val}(j) * B(i) \\
  & \quad \text{enddo} \\
  & \text{enddo}
\end{align*}
\]

\[
I^{DP}_{CRS} = \frac{2}{8 + 4 + 16\alpha + 12/N_{nzw} \text{ byte}} \ \text{flops} 
\]

- **New Parallelization Approach:**
  
  “Recursive Algebraic Coloring Engine” (RACE)

  by C.L. Alappatt (FAU)

- Publication in preparation
A quick walk-through
ECM model components: Data transfer times

- Optimistic transfer times through mem hierarchy
  \[ T_i = \frac{v_i}{b_i} \]
- Transfer time notation for some given loop kernel:
  \( \{T_{L1L2} | T_{L2L3} | T_{L3Mem}\} = \{4 \mid 8 \mid 18.4\} \text{ cy/8 iter} \)
- Input:
  - Cache properties
  - Application data transfer prediction
ECM model components: In-core execution

Best case: max throughput

\[ T_{\text{core min}} = \max(T_{\text{nOL}}, T_{\text{OL}}) \]

Worst case: critical path

\[ T_{\text{core max}} = T^{\text{CP}} \]

\( T_{\text{nOL}} \) interacts with cache hierarchy, \( T_{\text{OL}} \) does not
ECM model components: Overlap assumptions

- Notation for model contributions

\[
\{ T_{OL} \mid T_{nOL}|T_{L1L2}|T_{L2L3}|T_{L3Mem} \} = \{ 7 \mid 2 \mid 4 \mid 8 \mid 18.4 \} \text{cy/8 iter}
\]

- Most pessimistic overlap model: no overlap

\[
T_{ECM}^{Mem} = \max(T_{OL}, T_{nOL} + T_{L1L2} + T_{L2L3} + T_{L3Mem}) \text{ for in-mem data}
\]

Appropriate for most Intel Xeon CPUs up to and including Broadwell
ECM model:
Notation for runtime predictions

Example: no-overlap model

\[
\begin{align*}
\{ & \max(T_{OL}, T_{nOL}) \} \\
& \max(T_{OL}, T_{nOL} + T_{L1L2}) \\
& \max(T_{OL}, T_{nOL} + T_{L1L2} + T_{L2L3}) \\
& \max(T_{OL}, T_{nOL} + T_{L1L2} + T_{L2L3} + T_{L3Mem}) \}
\end{align*}
\]
ECM model: (Naive) saturation assumption

- Performance is assumed to scale across cores until a shared bandwidth bottleneck is hit.

\[ T_{ECM}(n) = \max\left(\frac{T_{ECM}^{Mem}}{n}, T_{L3Mem}\right) \implies n_S = \frac{T_{ECM}^{Mem}}{T_{L3Mem}} \]

- This is (sometimes) too optimistic near the saturation point. For improvements see J. Hofmann, G. Hager, and D. Fey: *On the accuracy and usefulness of analytic energy models for contemporary multicore processors*. Proc. ISC High Performance 2018. DOI: 10.1007/978-3-319-92040-5_2
MODELING A CONJUGATE-GRADIENT SOLVER

Building a model from components
A matrix-free CG solver

- 2D 5-pt FD Poisson problem
- Dirichlet BCs, matrix-free
- $N_x \times N_y = 40000 \times 1000$ grid
- CPU: Haswell E5-2695v3 CoD mode
Machine characteristics

- 2.3 GHz (fixed core & Uncore)
  - AVX2, 2 x FMA per cycle
  - 2 load & 1 store per cycle (in practice: 1+1)

- Cache characteristics
  - Inclusive, non-overlapping
  - $b_{L1L2} = 43 \text{ Byte/cy (gray)}$
  - $b_{L2L3} = 32 \text{ Byte/cy (white)}$

- Memory bandwidth per ccNUMA domain (saturated)
  - $b_{read} = 32.3 \text{ GByte/s} = 11.3 \text{ Byte/cy}$
  - $b_{copy} = 26.1 \text{ GByte/s} = 14.0 \text{ Byte/cy}$
**ECM model composition**

Naive implementation of all kernels (omp parallel for)

<table>
<thead>
<tr>
<th>while($\alpha_0 &lt; \text{tol}$):</th>
<th>$T_x$ [cy/8 iter]</th>
<th>$T_{E_{\text{Mem}}}$ [cy/8 iter]</th>
<th>$n_s$ [cores]</th>
<th>Full domain limit [cy/8 iter]</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\vec{v} = A\vec{p}$</td>
<td>${ 8 | 4 | 6.7 | 10 | 16.9 }$</td>
<td>37.6</td>
<td>3</td>
<td>16.9</td>
</tr>
<tr>
<td>$\lambda = \alpha_0 / \langle \vec{v}, \vec{p} \rangle$</td>
<td>${ 2 | 2 | 2.7 | 4 | 9.1 }$</td>
<td>17.8</td>
<td>2</td>
<td>9.11</td>
</tr>
<tr>
<td>$\vec{x} = \vec{x} + \lambda \vec{p}$</td>
<td>${ 2 | 4 | 6 | 16.9 }$</td>
<td>29.0</td>
<td>2</td>
<td>16.9</td>
</tr>
<tr>
<td>$\vec{r} = \vec{r} - \lambda \vec{v}$</td>
<td>${ 2 | 4 | 6 | 16.9 }$</td>
<td>29.0</td>
<td>2</td>
<td>16.9</td>
</tr>
<tr>
<td>$\alpha_1 = \langle \vec{r}, \vec{r} \rangle$</td>
<td>${ 2 | 2 | 1.3 | 2 | 4.6 }$</td>
<td>9.90</td>
<td>3</td>
<td>4.56</td>
</tr>
<tr>
<td>$\vec{p} = \vec{r} + \frac{\alpha_1}{\alpha_0} \vec{p}$, $\alpha_0 = \alpha_1$</td>
<td>${ 2 | 4 | 6 | 16.9 }$</td>
<td>29.0</td>
<td>2</td>
<td>16.9</td>
</tr>
<tr>
<td>Sum</td>
<td>152</td>
<td></td>
<td>81.3</td>
<td></td>
</tr>
</tbody>
</table>
CG performance – 1 core to full socket

- Multi-loop code well represented
- Single core performance predicted with 5% error
- Saturated performance predicted with < 0.5% error
- Saturation point predicted approximately
- Can be fixed by improved ECM model
Does the OpenMP barrier impact the performance?

At which problem size can this be expected?

Overhead (1 NUMA LD):

6 x 2000 cy

Time:

10 x 4x10^8 cy

→ No problem for any out of cache data set
CG with GS preconditioner: Naïve parallelization

Pipeline parallel processing: OpenMP barrier after each wavefront update (ugh!)
CG with GS preconditioner: additional kernels

<table>
<thead>
<tr>
<th>Non-PC model</th>
<th>$T_x$ [cy/8 iter]</th>
<th>$T_{ECM_{Mem}}$ [cy/8 iter]</th>
<th>$n_s$ [cores]</th>
<th>Full domain limit [cy/8 iter]</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\vec{z} = P\vec{r}$ (fw)</td>
<td>108</td>
<td>108</td>
<td>7</td>
<td>16.9</td>
</tr>
<tr>
<td>$\vec{z} = P\vec{r}$ (bw)</td>
<td>138</td>
<td>138</td>
<td>13</td>
<td>19.7</td>
</tr>
<tr>
<td>$\alpha = \langle \vec{r}, \vec{z} \rangle$</td>
<td>2</td>
<td>17.8</td>
<td>2</td>
<td>9.1</td>
</tr>
<tr>
<td><strong>Sum</strong></td>
<td><strong>416</strong></td>
<td><strong>127</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Back substitution does not saturate the memory bandwidth!
  - → full algorithm does not fully saturate
- Impact of barrier still negligible overall, but noticeable in the preconditioner
PCG measurement

- <2% model error for single threaded and saturated performance

- Expected large impact of barrier at smaller problem sizes in x direction
Conclusions

- Analytic modeling is worth the effort
  - Even if it’s inaccurate

- Even Roofline can yield amazing insights

- Analytic modeling
  - is not just for “simple kernels”
  - is composable
  - can cover programming model overhead, too
- … and yes, it is real work.

- No, it does not always work.

http://tiny.cc/kerncraft

Automatic Roofline/ECM modeling tool