The HPC Challenge Benchmarks and the PMaC project

- Certificates of relevance for benchmarks
  - Do they cover a useful performance space?
  - Do they enable reasoning about expected application performance?
- How practically to measure memory access patterns in nature
- Useful performance taxonomy

Components of a Performance Prediction Framework

- *Machine Profile* - characterizations of the rates at which a machine can (or is projected to) carry out fundamental operations abstract from the particular application
- *Application Signature* - detailed summaries of the fundamental operations to be carried out by the application independent of any particular machine

Combine Machine Profile and Application Signature using:
- *Convolution Method* - algebraic mapping of the application signature onto the machine profile to calculate a performance prediction
MAPS Data

MAPS – Memory bandwidth benchmark measures memory rates (MB/s) for different levels of cache (L1, L2, L3, Main Memory) and different access patterns (stride-one and random).

Convolutions

MetaSim trace collected on Cobalt60 simulating SC45 memory structure

<table>
<thead>
<tr>
<th>Basic-Block Number</th>
<th>Procedure Name</th>
<th># Memory References</th>
<th>L1 hit rate</th>
<th>L2 hit rate</th>
<th>Random ratio</th>
<th>Memory Bandwidth</th>
</tr>
</thead>
<tbody>
<tr>
<td>5247</td>
<td>Wallst</td>
<td>2.22E11</td>
<td>97.28</td>
<td>99.99</td>
<td>0.00</td>
<td>8851</td>
</tr>
<tr>
<td>10729</td>
<td>Poorgrd</td>
<td>4.90E08</td>
<td>88.97</td>
<td>92.29</td>
<td>0.20</td>
<td>1327</td>
</tr>
<tr>
<td>8649</td>
<td>Ucm6</td>
<td>1.81E10</td>
<td>92.01</td>
<td>97.07</td>
<td>0.23</td>
<td>572</td>
</tr>
</tbody>
</table>

Memory time = \( \sum_{i=1}^{n} \frac{\text{MemOpsBB}_i}{\text{MemRateBB}} \)

Memory Bandwidth (MB/s) vs. Size (0-byte words)
Results-Predictions for AVUS (Cobalt60)

<table>
<thead>
<tr>
<th>System</th>
<th>Actual time (s)</th>
<th>Predicted time (s)</th>
<th>% Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAVO IBM PWR3 (Habu)</td>
<td>8,601</td>
<td>11,180</td>
<td>+30%</td>
</tr>
<tr>
<td>ARL IBM PWR3 (Brainerd)</td>
<td>10,675</td>
<td>10,385</td>
<td>-3%</td>
</tr>
<tr>
<td>MHPCC IBM PWR3 (Tempest)</td>
<td>8,354</td>
<td>9,488</td>
<td>+14%</td>
</tr>
<tr>
<td>MHPCC IBM PWR4 (Hurricane)</td>
<td>4,932</td>
<td>4,258</td>
<td>-14%</td>
</tr>
<tr>
<td>NAVO IBM PWR4 (Marcellus)</td>
<td>4,375</td>
<td>4,445</td>
<td>+2%</td>
</tr>
<tr>
<td>ARL IBM PWR4 (Shelton)</td>
<td></td>
<td>6,192</td>
<td></td>
</tr>
<tr>
<td>NAVO IBM PWR4+ (Romulus)</td>
<td>3,272</td>
<td>3,239</td>
<td>-1%</td>
</tr>
<tr>
<td>ASC HP SC45</td>
<td>3,334</td>
<td>2,688</td>
<td>-19%</td>
</tr>
<tr>
<td>ARL Linux Networx Xeon Cluster</td>
<td></td>
<td>3,459</td>
<td></td>
</tr>
</tbody>
</table>

Spatial and Temporal Locality

How could one Quantify the Spatial and Temporal Locality in a Real Code?

\[
\text{SpatialScore}(N) = \sum_{i=1}^{N} \frac{(\text{Refs Stride } i / i)}{\text{Total Refs}}
\]

\[
\text{TemporalScore}(N) = \frac{\text{Observed Reuse}}{(\text{Total Refs} - \text{Spatial Refs})}
\]
It’s Harder Than it Looks

Where does one plot RandomAccess?

```
for (i = 0; i < N; i++) {
    add = random_number;
    table[add] ^= random_number;
}
```

Update (design goal)
Load + Store (temporal)
Load + Store (spatial)
Two loads + Store

HPC Challenge Benchmarks on axes of spatial and temporal locality