Parallel Programming Models

**Shared Memory**
- No sense of data locality — Very easy to use
- OS can map any virtual address to any physical address — Very inefficient

**Message Passing**
- Data locality is enforced — Difficult to use
- Programmer controls data traffic — High performance

**Partitioned Global Address Space (PGAS)**
- Flat memory — Easy to use
- Language is locality-aware — Automatic optimizations
- Programmer can exploit locality — Manual optimizations

---

**Stage 1 Performance Analysis**

**Motivation**
To analyze and compare different kinds of PGAS access performances in Chapel.

**Results**
<table>
<thead>
<tr>
<th>Access Type</th>
<th>Normalized Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local</td>
<td>1.0x</td>
</tr>
<tr>
<td>Local Shared</td>
<td>3.5x</td>
</tr>
<tr>
<td>Opt</td>
<td>1.0x</td>
</tr>
<tr>
<td>Remote</td>
<td>7.0x</td>
</tr>
</tbody>
</table>

---

**Stage 2 Hand Optimizations**

**Optimizations**
- No hand optimizations
- Reorganize loops to be able to use local block
- Manually localize remote data

**Results**

<table>
<thead>
<tr>
<th>Version</th>
<th>Speedup</th>
<th>Lines of Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>O0</td>
<td>1.1x-6.1x</td>
<td>4x-26x</td>
</tr>
<tr>
<td>O2</td>
<td>1.7x-60.1x</td>
<td>2.5x-11x</td>
</tr>
</tbody>
</table>

---

**Stage 3 Runtime System Support**

**Motivation**
What if programming language supported user-driven prefetching?

**Design**
- Productive
- Extensible
- Scalable
- Efficient
- Consistent

**Results**
- Promising performance improvements with very little modification on application code
- Similar performance to manual prefetching
- Almost all experiments required adding few calls to prefetch functions

---

**What is Chapel?**

Chapel is an emerging, productive, parallel programming language with the PGAS memory model. Unlike other PGAS languages and libraries, Chapel is standalone and is designed from scratch for parallel programming. Being a PGAS language, Chapel carries some inherent overheads of the easy-to-use memory model.

---

**References**


[3] *This work is a collaboration with Chapel team at Cray.*

engin@gwu.edu