Lithe: Enabling Efficient Composition of Parallel Libraries

Heidi Pan, Benjamin Hindman, Krste Asanović

xoxo@mit.edu • {benh, krste}@eecs.berkeley.edu
Massachusetts Institute of Technology • UC Berkeley

ParLab Boot Camp • August 19, 2009
Real-World Parallel Composition Example

Sparse QR Factorization
(Tim Davis, Univ of Florida)

System Stack

Software Architecture

SPQR
TBB
OpenMP
MKL
OS
Hardware
Out-of-the-Box Performance

Performance of SPQR on 16-core Machine

- **Out-of-the-Box**

![Graph showing time (sec) for different input matrices: landmark, deltaX, ESOC, and Rucci.](image)
Out-of-the-Box Libraries Oversubscribe the Resources

TBB | OpenMP
---|---
OS  | virtualized kernel threads
Hardware
MKL Quick Fix

Using Intel MKL with Threaded Applications
http://www.intel.com/support/performancetools/libraries/mkl/sb/CS-017177.htm

• If more than one thread calls Intel MKL and the function being called is threaded, it is important that threading in Intel MKL be turned off. Set `OMP_NUM_THREADS=1` in the environment.
Sequential MKL in SPQR
Sequential MKL Performance

Performance of SPQR on 16-core Machine

- Out-of-the-Box
- Sequential MKL

Input Matrix

<table>
<thead>
<tr>
<th></th>
<th>landmark</th>
<th>deltaX</th>
<th>ESOC</th>
<th>Rucci</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time (sec)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>2.0</td>
<td>85</td>
<td>1200</td>
</tr>
<tr>
<td></td>
<td>1.0</td>
<td>15.0</td>
<td>75</td>
<td>1000</td>
</tr>
<tr>
<td></td>
<td>1.5</td>
<td>20.0</td>
<td>65</td>
<td>800</td>
</tr>
<tr>
<td></td>
<td>2.0</td>
<td>25.0</td>
<td>60</td>
<td>400</td>
</tr>
<tr>
<td></td>
<td>2.5</td>
<td>30.0</td>
<td></td>
<td>200</td>
</tr>
<tr>
<td></td>
<td>3.0</td>
<td>35.0</td>
<td></td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>3.5</td>
<td>40.0</td>
<td></td>
<td>50</td>
</tr>
</tbody>
</table>
SPQR Wants to Use Parallel MKL

No \textit{task-level} parallelism!

Want to exploit \textit{matrix-level} parallelism.
Tim Davis manually tunes libraries to effectively partition the resources.
Manually Tuned Performance

Performance of SPQR on 16-core Machine

Out-of-the-Box
Sequential MKL
Manually Tuned

Input Matrix

Time (sec)
Manual Tuning Destroys Black Box Abstractions

Tim Davis

OMP_NUM_THREADS = 4

MKL
OpenMP
Manual Tuning Destroys Code Reuse and Modular Updates
Virtualized Threads are Bad

Different codes compete unproductively for resources.
Harts: Hardware Thread Contexts

- Represent real hw resources.
- Requested, not created.
- OS doesn’t manage harts for app.

Diagram:
- Core 0 to Core 7
- TBB, MKL, OpenMP
- SPQR
- OS

Harts
Diagram showing the sharing of Harts (Hardware threads) over time. The diagram includes layers labeled TBB, OpenMP, OS, and Hardware, indicating different levels of the software stack. The Harts (Hart 0, Hart 1, Hart 2, Hart 3) are distributed across these layers and are partitioned over time.
Hierarchical Cooperative Scheduling

Hierarchical, Cooperative

TBB Sched

Direct Control of Resources

OS (Harts)

Hardware

Column Elimination Tree

LAPACK

MKL

OpenMP Sched
Standard Lithe ABI

- Analogous to function call ABI for enabling interoperable codes.
- Mechanism for sharing harts, not policy.

TBB\textsubscript{Lithe} Scheduler

| enter | yield | request | register | unregister |

interface for sharing harts

OpenMP\textsubscript{Lithe} Scheduler

| enter | yield | request | register | unregister |

interface for exchanging values

Caller

| call | return |

Callee

| call | return |
SPQR with Lithe

Diagram showing the interaction between SPQR, TBB, MKL, and OpenMP with Lithe.
SPQR with Lithe

- TBB
- MKL
- OpenMP
- Lithe

Call
Req
Ret
Performance of SPQR with Lithe

Input Matrix

Out-of-the-Box  | Manually Tuned  | Lithe

Time (sec)

landmark  | deltaX  | ESOC  | Rucci
Lithe: Enabling Efficient Composition of Parallel Libraries
Acknowledgements

We would like to thank George Necula and the rest of Berkeley Par Lab for their feedback on this work.

Research supported by Microsoft (Award #024263) and Intel (Award #024894) funding and by matching funding by U.C. Discovery (Award #DIG07-10227). This work has also been in part supported by a National Science Foundation Graduate Research Fellowship. Any opinions, findings, conclusions, or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation. The authors also acknowledge the support of the Gigascale Systems Research Focus Center, one of five research centers funded under the Focus Center Research Program, a Semiconductor Research Corporation program.