MPI+X Programming Models on Future Systems – the Search for Lowest-Order Effects

Georg Hager
Erlangen Regional Computing Center (RRZE)

Programming Models on the Road to Exascale
ISC High Performance 2015
July 13, 2015, Frankfurt, Germany
Outline

- Resource-aware software engineering
  - Hardware bottlenecks
  - What we need and what we get – resource balance (Kung)
  - Lowest-order thinking (excavator aerodynamics)

- MPI+X programming models
  - $X = \{\}$, threading, accelerator
  - Opportunities for addressing the lowest order
  - How to find the lowest order?
Resource-aware software engineering

Resources are means to an end
Resource-aware software engineering

Resources are means to an end

Computation
Resource-aware software engineering

Resources are means to an end

- Computation
- Pipeline throughput
- Memory transfer
- Superscalarity
- Inter-LD transfer
- PCIe transfer
- Inter-cache transfer
- Disk I/O
- Network communication
Resource-aware software engineering

Resources are means to an end

- Computation
- Pipeline throughput
- Memory transfer
- Superscalarity
- Inter-LD transfer
- PCIe transfer
- Inter-cache transfer
- Disk I/O
- Network communication

Solution
Resource balance: what we need and what we get

Initial idea: code balance vs. machine balance

\[ B_m = \frac{b_s}{P_{\text{peak}}} \quad \text{bandwidth} \]

\[ B_c = \frac{V}{C} \quad \text{data volume} \]

\[ \text{computation (work)} \]
Resource balance: what we need and what we get

Initial idea: code balance vs. machine balance

$B_m = \frac{b_S}{P_{\text{peak}}}$

$B_c = \frac{V}{C}$

what we need

what we get

Optimal co-design if what we need is what we get

H.T. Kung: Memory requirements for balanced computer architectures. Proc. ISCA’86, DOI: 10.1145/17356.17362
Generalization of the balance concept: Lowest order

Limited resources impose upper (lower) performance (runtime) limits

- Computation
  - Pipeline throughput
  - Memory transfer
    - Superscalarity
    - Inter-cache transfer
    - Inter-LD transfer
    - PCIe transfer
    - Network communication
    - Disk I/O

The bottleneck
Generalization of the balance concept: Lowest order

Limited resources impose upper (lower) performance (runtime) limits

Computation
- Pipeline throughput
- Memory transfer
- Superscalarity
- Inter-cache transfer
- Inter-LD transfer
- PCIe transfer
- Network communication
- Disk I/O

The bottleneck

Callahan, Cocke, Kennedy (1988), DOI: 10.1016/0743-7315(88)90002-0

Williams, Waterman, Patterson (2009), DOI: 10.1145/1498765.1498785
Reality: Next to lowest order

Simple balance picture does not hold due to non-overlap

Treibig & Hager (2010), DOI: 10.1007/978-3-642-14390-8_64

Stengel, Treibig, Hager, Wellein (2015), DOI: 10.1145/2751205.2751240
Reality: Next to lowest order

Simple balance picture does not hold due to non-overlap

Treibig & Hager (2010),
DOI: 10.1007/978-3-642-14390-8_64

Stengel, Treibig, Hager, Wellein (2015),
DOI: 10.1145/2751205.2751240
Reality: Next to lowest order

Simple balance picture does not hold due to non-overlap

Treibig & Hager (2010), DOI: 10.1007/978-3-642-14390-8_64

Stengel, Treibig, Hager, Wellein (2015), DOI: 10.1145/2751205.2751240
Reality: Next to lowest order

Simple balance picture does not hold due to non-overlap

Treibig & Hager (2010), DOI: 10.1007/978-3-642-14390-8_64

Stengel, Treibig, Hager, Wellein (2015), DOI: 10.1145/2751205.2751240
Reality: Next to lowest order

Simple balance picture does not hold due to non-overlap

Treibig & Hager (2010), DOI: 10.1007/978-3-642-14390-8_64
Stengel, Treibig, Hager, Wellein (2015), DOI: 10.1145/2751205.2751240
Getting to lowest order: a software challenge

Potential of overlap is limited by the minimum requirements of the software w.r.t. the hardware
Getting to lowest order: a software challenge

Potential of overlap is limited by the minimum requirements of the software w.r.t. the hardware

- Computation
  - Memory transfer
  - Inter-cache transfer
- Network communication
- Disk I/O
Getting to lowest order: a software challenge

Potential of overlap is limited by the minimum requirements of the software w.r.t. the hardware.
Getting to lowest order: a software challenge

Potential of overlap *is* limited by the minimum requirements of the software w.r.t. the hardware

... and it *should be* limited/guided by lowest-order thinking!
Getting to lowest order: a software challenge

Potential of overlap *is* limited by the minimum requirements of the software w.r.t. the hardware.

... and it *should be* limited/guided by lowest-order thinking!

**Resource optimization ==**
Getting to lowest order: a software challenge

Potential of overlap *is* limited by the minimum requirements of the software w.r.t. the hardware

... and it *should be* limited/guided by lowest-order thinking!

Resource optimization ==

- exposing the lowest-order bottleneck
Getting to lowest order: a software challenge

Potential of overlap is limited by the minimum requirements of the software w.r.t. the hardware

... and it should be limited/guided by lowest-order thinking!

Resource optimization =

- exposing the lowest-order bottleneck
- reducing the impact of the bottleneck
Typical dead end: excavator aerodynamics

Getting to lowest order is only useful if it promises a significant return
Typical dead end: excavator aerodynamics

Getting to lowest order is only useful if it promises a significant return
Typical dead end: excavator aerodynamics

Getting to lowest order is only useful if it promises a significant return
Typical dead end: excavator aerodynamics

Getting to lowest order is only useful if it promises a significant return
Typical dead end: excavator aerodynamics

Getting to lowest order is only useful if it promises a significant return

... even if your programming model allows it!
Now what about the X?

X =
Now what about the X?

\[ X = \{ \} \]

OpenMP, TBB, OmpSs, pthreads, Cilk(+)  
CUDA, OpenCL  
OpenACC, OpenMP4  
some-library-that-does-the-trick  
tiny.cc/GHOST
Opportunities for exposing the lowest order
Opportunities for exposing the lowest order

1. If a programming model (i.e., some X) lets me expose the lowest order, I'm fine with it
Opportunities for exposing the lowest order

1. If a programming model (i.e., some X) lets me expose the lowest order, I’m fine with it
2. If some X allows me to expose the lowest order better than any other, it may be a better candidate

Examples:
Opportunities for exposing the lowest order

1. If a programming model (i.e., some X) lets me expose the lowest order, I'm fine with it
2. If some X allows me to expose the lowest order better than any other, it may be a better candidate

Examples:

OMP tasking for comm./comp. overlap
Opportunities for exposing the lowest order

1. If a programming model (i.e., some X) lets me expose the lowest order, I’m fine with it
2. If some X allows me to expose the lowest order better than any other, it may be a better candidate

Examples:

- OMP tasking for comm./comp. overlap
- DSL for exposing data parallelism and data flow in stencil algorithms
Opportunities for exposing the lowest order

1. If a programming model (i.e., some X) lets me expose the lowest order, I’m fine with it
2. If some X allows me to expose the lowest order better than any other, it may be a better candidate

Examples:

- OMP tasking for comm./comp. overlap
- OmpSs for extracting the critical path
- DSL for exposing data parallelism and data flow in stencil algorithms
Opportunities for exposing the lowest order

1. If a programming model (i.e., some X) lets me expose the lowest order, I’m fine with it
2. If some X allows me to expose the lowest order better than any other, it may be a better candidate

Examples:

- **OMP tasking for comm./comp. overlap**
- **OmpSs for extracting the critical path**
- **DSL for exposing data parallelism and data flow in stencil algorithms**
- **MPI-3 shared memory for reducing intra-node MPI overhead**
Opportunities for exposing the lowest order

1. If a programming model (i.e., some X) lets me expose the lowest order, I’m fine with it
2. If some X allows me to expose the lowest order better than any other, it may be a better candidate

Examples:

- OMP tasking for comm./comp. overlap
- OmpSs for extracting the critical path
- DSL for exposing data parallelism and data flow in stencil algorithms
- MPI-3 shared memory for reducing intra-node MPI overhead
- GHOST for expressing asynchronicity and enforcing resource affinity
Finally:
How do you know what the lowest order is?
Finally:
How do you know what the lowest order is?

- Profiling? Hardware counter measurements?
  Automatic tuning advice?

File foo.cc, line 56: Loop shows 50.0% L1 cache hit rate – consider optimization
Finally:
How do you know what the lowest order is?

- Profiling? Hardware counter measurements? Automatic tuning advice?

File foo.cc, line 56: Loop shows 50.0% L1 cache hit rate – consider optimization

- Performance modeling of hardware-software interaction!
Finally:
How do you know what the lowest order is?

- Profiling? Hardware counter measurements? Automatic tuning advice?

File foo.cc, line 56: Loop shows 50.0% L1 cache hit rate – consider optimization

- Performance modeling of hardware-software interaction!
  - Roofline model, ECM model, LogP model, …
Finally:
How do you know what the lowest order is?

- Profiling? Hardware counter measurements? Automatic tuning advice?

File foo.cc, line 56: Loop shows 50.0% L1 cache hit rate – consider optimization

- Performance modeling of hardware-software interaction!
  - Roofline model, ECM model, LogP model, …
  - Performance patterns (Treibig Hager, Wellein (2012), DOI: 10.1007/978-3-642-36949-0_50)

Visit our ISC15 workshop
Performance Modeling: Methods & Applications
(Marriott, Room Gold 1+2)
Take-home messages

- If it does the trick, it is a candidate
  - The trick being the full utilization of a bottleneck

- If it does the trick better than anything else, it may be worth serious consideration

- If it is sustainable, take it.

- What is the trick?

  → A performance model will probably guide you!
Thank You.