Employing MPI Collectives for Timing Analysis on Embedded Multi-Cores

Martin Frieb, Alexander Stegmeier,
Jörg Mische, Theo Ungerer

Department of Computer Science
University of Augsburg

16th International Workshop on
Worst-Case Execution Time Analysis
July 5, 2016

Motivation

Björn Lisper, WCET 2012:
"Towards Parallel Programming Models for Predictability"

- Shared memory does not scale
  ⇒ Replace it with distributed memory
- Replace bus with Network-on-Chip (NoC)
- Learn from Parallel Programming Models
- e.g. Bulk Synchronous Programming (BSP):
  Execute program in *supersteps*:
  1. Local computation
  2. Global communication
  3. Barrier

MPI programs

Similar programming model comes with MPI programs

- At a *collective operation*, all (or a group of) cores work together
- local computation, followed by communication
  ⇒ implicit barrier
- One core for coordination and distribution (master), others for computation (slave)
- Examples:
  - Barrier
  - Broadcast
  - Global sum

Outline

Background

Timing Analysis of MPI Collective Operations

Case Study: Timing Analysis of the CG Benchmark

Summary and Outlook

Outline

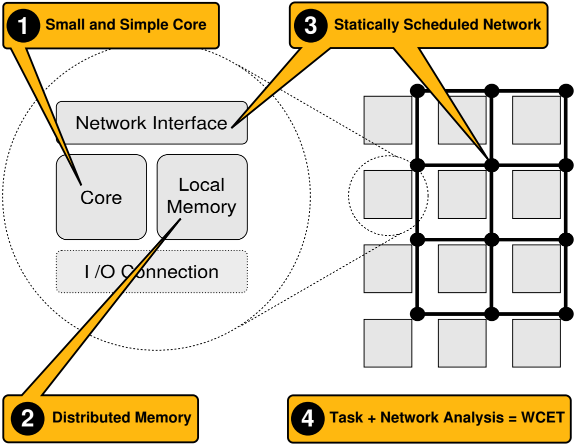
Background

Timing Analysis of MPI Collective Operations

Case Study: Timing Analysis of the CG Benchmark

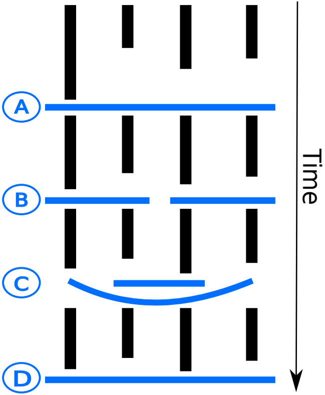
Summary and Outlook

Underlying Architecture



[Metzlaff et al.: A Real-Time Capable Many-Core Model, RTSS-WiP 2012]

Structure of a MPI program



Same sequential code on all cores

(A) Barrier after initialization

(B) Data exchange

(C) Data exchange

(D) Global operation

Outline

Background

Timing Analysis of MPI Collective Operations

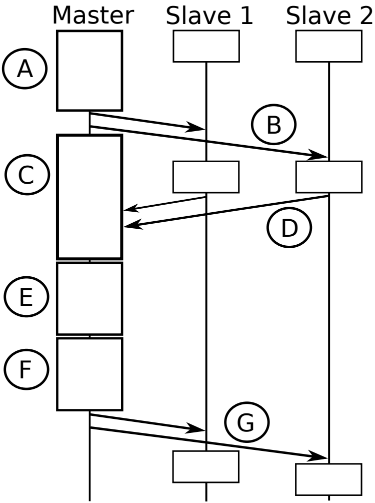
Case Study: Timing Analysis of the CG Benchmark

Summary and Outlook

Structure of MPI_Allreduce

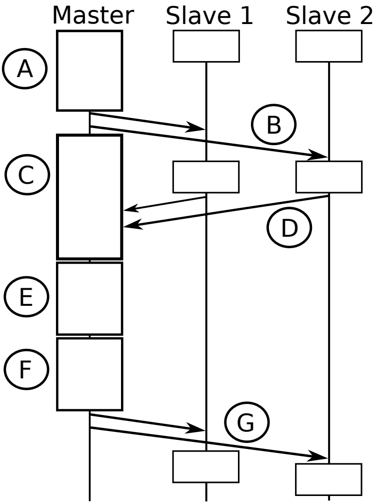
- Global reduction operation
- Broadcasts result afterwards

Structure of MPI_Allreduce



- Global reduction operation
- Broadcasts result afterwards

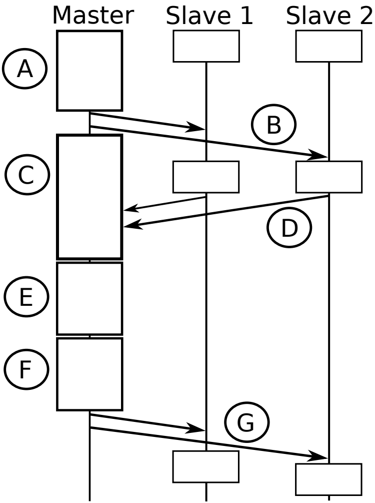
Structure of MPI_Allreduce



- Global reduction operation
- Broadcasts result afterwards

(A) Initialization

Structure of MPI_Allreduce

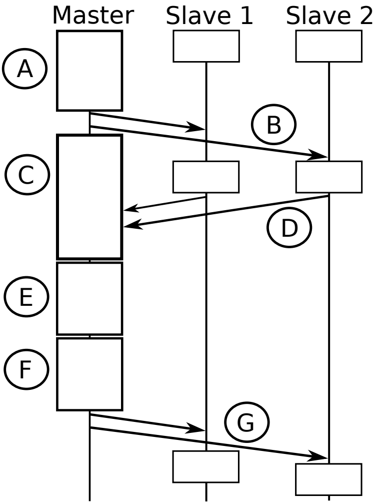


- Global reduction operation
- Broadcasts result afterwards

(A) Initialization

(B) Acknowledgement

Structure of MPI_Allreduce



- – Global reduction operation
- – Broadcasts result afterwards

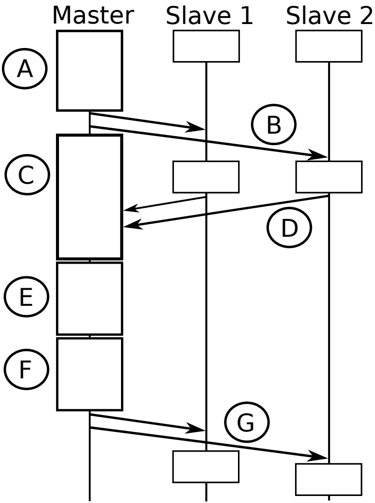
(A) Initialization

(B) Acknowledgement

(C) Data structure initialization

(D) Send values

Structure of MPI_Allreduce



- Global reduction operation
- Broadcasts result afterwards

(A) Initialization

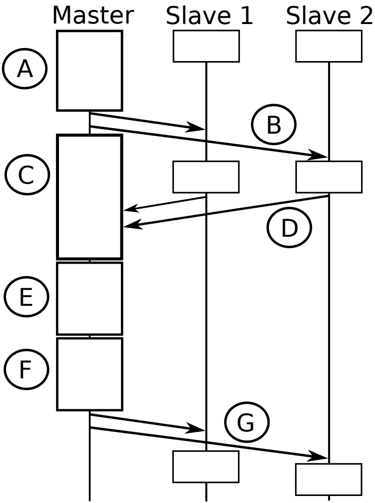
(B) Acknowledgement

(C) Data structure initialization

(D) Send values

(E) Collect and store values

Structure of MPI_Allreduce



- – Global reduction operation
- – Broadcasts result afterwards

(A) Initialization

(B) Acknowledgement

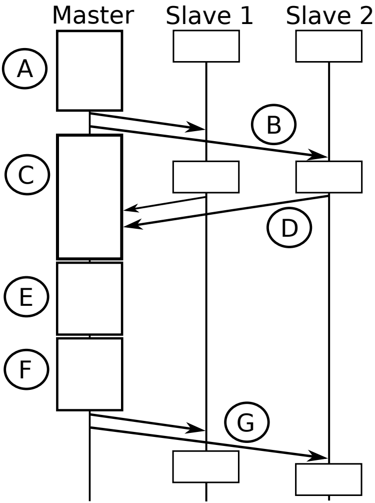
(C) Data structure initialization

(D) Send values

(E) Collect and store values

(F) Apply global operation

Structure of MPI_Allreduce



- Global reduction operation
- Broadcasts result afterwards

(A) Initialization

(B) Acknowledgement

(C) Data structure initialization

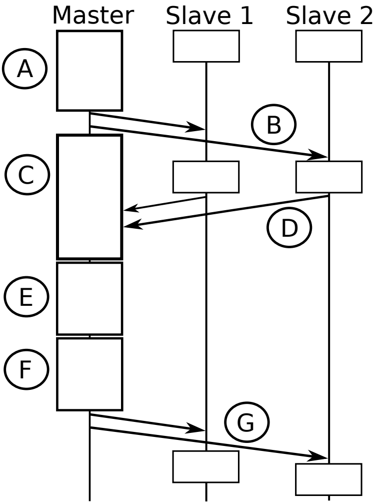
(D) Send values

(E) Collect and store values

(F) Apply global operation

(G) Broadcast result

Structure of MPI_Allreduce



- – Global reduction operation
- – Broadcasts result afterwards

(A) Initialization

(B) Acknowledgement

(C) Data structure initialization

(D) Send values

(E) Collect and store values

(F) Apply global operation

(G) Broadcast result

WCET $= \Sigma$ A to G

Analysis of MPI_Allreduce

- WCET of sequential parts estimated with OTAWA
- Worst-case traversal time (WCTT) of communication parts has to be added
- Result: Equation with parameters
  - #values to be transmitted
  - #communication partners
  - Dimensions of NoC
  - Transportation times
  - Time between Core and NoC
- Equation can be reused for *any* application on same architecture

Analysis of MPI_Sendrecv

Analysis of MPI_Sendrecv

- Purpose: simultaneous send and receive to avoid deadlock
- Often used for data exchange: pass data to next core

Analysis of MPI_Sendrecv

- Purpose: simultaneous send and receive to avoid deadlock
- Often used for data exchange: pass data to next core
- Structure:
  - Initialization

Analysis of MPI_Sendrecv

- Purpose: simultaneous send and receive to avoid deadlock
- Often used for data exchange: pass data to next core
- Structure:
  - Initialization
  - Acknowledgement

Analysis of MPI_Sendrecv

- Purpose: simultaneous send and receive to avoid deadlock
- Often used for data exchange: pass data to next core
- Structure:
  - Initialization
  - Acknowledgement
  - Sending and receiving of values

Analysis of MPI_Sendrecv

- Purpose: simultaneous send and receive to avoid deadlock
- Often used for data exchange: pass data to next core
- Structure:
  - Initialization
  - Acknowledgement
  - Sending and receiving of values
- Result: Equation with parameters
  - #values to be transmitted
  - Transportation times
  - Time between Core and NoC

Analysis of MPI_Sendrecv

- Purpose: simultaneous send and receive to avoid deadlock
- Often used for data exchange: pass data to next core
- Structure:
  - Initialization
  - Acknowledgement
  - Sending and receiving of values
- Result: Equation with parameters
  - #values to be transmitted
  - Transportation times
  - Time between Core and NoC
- Equation can be reused for *any* application on same architecture

Outline

Background

Timing Analysis of MPI Collective Operations

Case Study: Timing Analysis of the CG Benchmark

Summary and Outlook

The CG Benchmark

- Conjugate Gradient method from mathematics
- Optimization method to find the minimum/maximum of a multidimensional function
  - Operations on a large matrix
  - Distributed on several cores
  - Cores exchange data a number of times
- Taken from NAS Parallel Benchmark Suite for highly parallel systems
- Adapted for C + MPI

Setting for the analysis

- Simple ARM cores with 5-stage pipeline
- Distributed memory, no caches, 10 cycles memory access latency
- 4x4 PaterNoster NoC, arranged as unidirectional torus
- Sending and receiving takes 3 assembler instructions $+$ 4 cycles from Pipeline to NoC
- Due to the usage of time division multiplexing (TDM), a WCTT can be estimated

  → Stegmeier et al.: WCTT bounds for MPI Primitives in the PaterNoster NoC, *14th International Workshop on Real-Time Networks (RTN)*, July 5th, 2016, Toulouse

Analysis of the CG Benchmark

- Initialization not analysed
- Structure of one benchmark iteration:
  - Alternating sequential and communication parts
  - Some of them are repeated for 15 times in a `for` loop
- Summation of parts gives equation
  - Shortcoming: ignored pipeline states at summation

$$\begin{aligned} WCET_{cg} &= 1\,896\,959 \\ &+ WCET_{AR}(2, 15) + 17 \cdot WCET_{AR}(1, 3) \\ &+ 16 \cdot (WCET_{AR}(351, 3) + WCET_{SR}(351)) \end{aligned}$$

Results

Specific numbers:

- $WCET_{AR}(2, 15) = 8.158\ cycles$
- $WCET_{AR}(1, 3) = 1.071\ cycles$
- $WCET_{AR}(351, 3) = 113.073\ cycles$
- $WCET_{SR}(351) = 11.396\ cycles$

$$\begin{aligned} WCET_{cg} &= 1\,896\,959 \\ &+ 8\,158 + 17 \cdot 1\,071 \\ &+ 16 \cdot (113\,073 + 11\,396) \\ &= 3\,914\,828\ cycles \end{aligned}$$

Outline

Background

Timing Analysis of MPI Collective Operations

Case Study: Timing Analysis of the CG Benchmark

Summary and Outlook

Summary and Outlook

- Learn from parallel programming models
- MPI collectives: clear separation of computation and communication phases

  ⇒ combine separate WCET estimates for sequential code and MPI collectives
- Analyzed in case study: MPI_Allreduce, MPI_Sendrecv, CG benchmark
- Results can be reused for any application on same architecture

Summary and Outlook

- Learn from parallel programming models
- MPI collectives: clear separation of computation and communication phases
  ⇒ combine separate WCET estimates for sequential code and MPI collectives
- Analyzed in case study: MPI_Allreduce, MPI_Sendrecv, CG benchmark
- Results can be reused for any application on same architecture

Outlook

- Improve MPI implementation: e. g. workload distribution
- Optimize hardware support

Questions?

Thank you for your attention!

Additional slides

Equation of MPI_Allreduce

- f: #values to be transmitted
- χ: #communication partners
- n: Dimensions of NoC
- $t_{transm,X}$: Transportation times
- t_{Buf}: Time from Core to NoC and backwards

$$\begin{aligned}
WCET_{AR}(f,\chi) =& 273 + 35f\chi + 141\chi \\
& + max(23 + 6n^2 + 11\chi, 24 + 2(t_{transm,\chi} + t_{Buf})) \\
& + (f-1)max(35\chi, t_{transm,\chi}) \\
& + (66 + t_{transm,\chi})f + t_{Buf}
\end{aligned}$$

Equation of MPI_Sendrecv

- f: #values to be transmitted
- $t_{transm,X}$: Transportation times
- t_{Buf}: Time from Core to NoC and backwards

$$WCET_{SR}(f) = 108 + 2 \cdot (t_{transm,1} + t_{Buf}) + max(f \cdot 32, t_{transm,f}) + t_{Buf}$$