Five Open Problems on Memory Models for Extreme-Scale Parallel Processing

Guang R. Gao
ACM Fellow and IEEE Fellow
Distinguished Professor, ECE
University of Delaware
And
Founder, ETI
Outline

• Introduction
• The Codelets Model (based on [Gao et. al, 04/11])
• Memory Models: Views from Two Perspectives
• Open Questions from Classical Memory Consistency Models
• Status and Plan
• A Wake Up Call
• Our Position Statement
• Conclusions
Abstract

Programming Models

Execution Model

Programming Environment Platforms

Execution Model API

Abstract Machine Models

Users

Users
Execution Model API

Abstract Machine Models

Machine Runtime System

Hardware Architecture

Programming Environment Platforms

High-Level Programming API (MPI, Open MP, CnC, Xio, Chapel, etc.)

Software packages
Program libraries
Utility applications

Compilation

Compliers

Tools/SDK

Users

Users

Programming Models

Execution Model

Language Runtime

Execution Model API
Our Position

• *Parallel execution/abstract machine model*

• *Language Runtime vs. Machine Runtime*

• *Delaware Codelet Model and Its Abstract Machines:*
  • SWARM
  • DART
Outline

• Introduction
• The Codelets Model (based on [Gao et. al, 04/11])
• Memory Models: Views from Two Perspectives
• Open Questions from Classical Memory Consistency Models
• Status and Plan
• A Wake Up Call
• Our Position Statement
• Conclusions
What is a Codelet?

• Intuitively:

  A unit of computation which interacts with the global state only at its entrance and exit points

• Terminology

  - I would be very cautious to use the term “functional programming” here – which usually means “stateless”!
  - But, I like the term of single-assignment and dataflow programming models
A Dynamic Multithreaded Execution Model and Abstract Machine

Best references:
For SGT/MGT: Kevin Theobald’s Ph.D Thesis [1999]
For LGT level: Juan Cuvillo’s Ph.D Thesis [2008]

The relation with classical Dennis’ static dataflow/abstract machine model - see [Dennis and Gao, Supercomputing88].
Outline

• Introduction
• The Codelets Model (based on [Gao et. al, 04/11])
• Memory Models: Views from Two Perspectives
• Open Questions from Classical Memory Consistency Models
• Status and Plan
• A Wake Up Call
• Our Position Statement
• Conclusions
What Is A Memory Model?
Perspectives of Memory Models
- A Tale of Two Cities

• Perspective 1 (classical): the view of shared memory presented to multiple processors executing Reads andWrites (The classical SMP view – a la Leslie Lamport 1978/1979)

• Perspective 2: "memory model" as the ideal interface to data and code objects by application programs? (The memory aspect of a PXM)
What is A Shared Memory Execution Model from the View of PXM?

**Thread Model**
A set of rules for creating, destroying and managing threads

**Execution Model**

**Memory Model**
Dictates the ordering of memory operations

**Synchronization Model**
Provides a set of mechanisms to protect from data races

---

**The Thread Abstract Machine**

---

9/3/14
Open-Questions-on-Memoty-Models
Vision of a massively parallel system

Many-Core Processing Chips

Main Memory: Associative Directories and DRAM

Archive Memory Level: Access Controllers and Flash

Switch

Switch

Switch
Two Essential Aspects of A Memory Model (from the classical perspective)

• Addressing Model
  Question: How is a (global) memory location addressed?

• Consistency Model
  Question: when multiple (concurrent) reads and writes to a memory location – what are the results of these operations?
The SC Memory Model

Memory

S = interleave (S₁, S₂, …Sn)

P₁

P₂

…

Pₙ
Timeline of Memory Models

- 1974: Dataflow
  
- 1978: Lamport Timestamps

- 1979: Sequential Consistency

- 1986: Release Consistency

- 1990: Weak Consistency

- 2000: Causal Acyclic Consistency

- 2005: Location Consistency

- 2008: C++ Memory Model

- 2010: The Java Memory Model

Open Questions on Memory Models
Outline

• Introduction
• The Codelets Model (based on [Gao et. al, 04/11])
• Memory Models: Views from Two Perspectives
• Open Questions from Classical Memory Consistency Models
• Status and Plan
• A Wake Up Call
• Our Position Statement
• Conclusions
Two More Open Questions: Questions Q1 and Q2?

**Q1**: Should the hardware (architecture) permit > 1 alternative paths of routing of the memory operations (transactions) along the way? Can one core be connected to one memory bank by multiple paths?

**Q2**: If the answer of Q1 is true (I assume it is) – then is it possible that the two operations arrive at their destination out-of-order?
Your Answers to the Following Questions Q1 and Q2?

**Q1:** Should the hardware (architecture) permit > 1 alternative paths of routing of the memory operations (transactions) along the way? Can one core be connected to one memory bank by multiple paths?

**Q2:** If the answer of Q1 is true (I assume it is) – then is it possible that the two operations arrive at their destination out-of-order?

<table>
<thead>
<tr>
<th>No.</th>
<th>Answer to Q1</th>
<th>Answer to Q2</th>
<th>Which one ?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>No</td>
<td>N/A</td>
<td></td>
</tr>
</tbody>
</table>
Q1: Scenario 1

| I0: ST X, 1 |
| I1: ST X, 2 |
| I2: LD R1,X |

**Scenario 1:**
There is only one path between the processor and the memory bank.
Q1: Scenario 1

**Scenario 1:**
There is only one path between the processor and the memory bank.

<table>
<thead>
<tr>
<th>I0: ST X, 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>I1: ST X, 2</td>
</tr>
<tr>
<td>I2: LD R1, X</td>
</tr>
</tbody>
</table>

Memory Location for X
Q1: Scenario 1

Scenario 1:
There is only one path between the processor and the memory bank.

I0: ST X, 1
I1: ST X, 2
I2: LD R1, X
Q1: Scenario 1

**Scenario 1:**
There is only one path between the processor and the memory bank.

I0: **ST X, 1**
I1: **ST X, 2**
I2: **LD R1, X**

Everything as expected!!!
Q1: Scenario 2

**Scenario 2:**
There are multiple paths between the processor and the memory bank.

I0: \textbf{ST} X, 1
I1: \textbf{ST} X, 2
I2: \textbf{LD} R1, X
Q1: Scenario 2

Scenario 2:
There are multiple paths between the processor and the memory bank.

I₀: ST X, 1
I₁: ST X, 2
I₂: LD R₁, X
Q1: Scenario 2

**Scenario 2:**
There are multiple paths between the processor and the memory bank.

- **I0:** \textcolor{red}{ST X, 1}
- **I1:** \textcolor{green}{ST X, 2}
- **I2:** \textcolor{red}{LD R1, X}

\textit{I1} is stuck here due to traffic in the network.
Q1: Scenario 2

**Scenario 2:**
There are multiple paths between the processor and the memory bank.

I0: ST X, 1  
I1: ST X, 2  
I2: LD R1, X

I1 is stuck here due to traffic in the network

Memory Location for X

R1 = 1 = Error !!!!
Possible Answers to the Questions Q1 and Q2

**Q1:** Should the hardware (architecture) permit > 1 alternative paths of routing of the memory operations (transactions) along the way? Can one core be connected to one memory bank by multiple paths?

**Q2:** If the answer of Q1 is true (I assume it is) – then is it possible that the two operations arrive at their destination out-of-order?

<table>
<thead>
<tr>
<th>No.</th>
<th>Answer to Q1</th>
<th>Answer to Q2</th>
<th>Who answered</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>Yes</td>
<td>GG, MM</td>
</tr>
<tr>
<td>2</td>
<td>Yes</td>
<td>No</td>
<td>BS</td>
</tr>
<tr>
<td>3</td>
<td>No</td>
<td>N/A</td>
<td>MS,DD,SS</td>
</tr>
</tbody>
</table>
Another Open Question : Q3

Q3: What happens when two (or more) concurrent load/store operations (arrive) at the same memory location at the same time?

Answers ?
The Coherence Barrier

Open Question (Q3): Can we go beyond the “Memory Coherence” barriers?

Your Answer: Yes ? or No ?
One More Open Question on Memory Model?

- **Question Q4:** Should a memory model preserves the notion of *causality*?
Yet Another Open Question (Q5)

Q5: Does the performance gain due to relaxed SC-derived model worth the added programming complexity?
Outline

• Introduction
• The Codelets Model (based on Gao et. al, 04/11)
• Memory Models: Views from Two Perspectives
• Open Questions from Classical Memory Consistency Models
• Status
• A Wake Up Call
• Our Position Statement
• Conclusions
A Summary

– The Open Problems have been identified and documented

– Impact of LC on caches and scratchpad memory has been studied under the Cyclops-64 many-core chip architecture.
Outline

• Introduction
• The Codelets Model (based on [Gao et. al, 04/11])
• Memory Models: Views from Two Perspectives
• Open Questions from Classical Memory Consistency Models
• Status
• A Wake Up Call
• Our Position Statement
• Conclusions
Remarks on Q1/Q2 - from A World Legend Computer Architect US Computer Industry

• Mr. X – a Cray-Award Winner – lately described Open Question Q1/Q2 he has been struggling with in a large-scale manycore chip for DoD.

• Mr. X told me: “… What I would need is some glimmer of evidence that the problem I described (i.e. Q1) can be solved. If it is hopeless then I would just give up. Note that I very seldom give up on anything, but this could be one of those cases….”
Observation on Q5: An Opinion from A Well-Respected Academia Colleague

• Open Question Q5: Does the performance gain due to relaxed SC-derived model worth the added programming complexity?

• Observation, 

No, not really!
A Wake Up Call

Houston: We Have A Problem!
It Is Time Look Beyond The Classical View of Memory Models

- A fresh look from the Perspective 2 (see page 14) based on program execution model (PXM) angle
- This means that we should look at the thread model, synchronization model, and memory model all together
Outline

• Introduction
• The Codelets Model (based on [Gao et. al, 04/11])
• Memory Models: Views from Two Perspectives
• Open Questions from Classical Memory Consistency Models
• Status
• A Wake Up Call and Our Position
• Our Position Statement
• Conclusions
Our Position Statement

High-performance computing systems for applications important in the future will need these principles:

- It is feasible to build a multi-core operating system (OS) that implements virtual memory, and honors the principles of modular software construction, using runtime software that implements a \textit{codelet} program execution model.

- Performance and energy efficiency can be enhanced through co-design of \textit{new architecture features} that replace resource management functions of runtime software with efficient hardware mechanisms.

- The resulting systems will offer benefits in \textit{programmability}, \textit{portability} and \textit{reusability} absent in current systems.
A Proposal

- **Obj-1**: Define a unified execution model based on codelet and codelet machine concepts (e.g. SWARM, FreshBreeze, DART, OCR, ParalleX, etc.) with a sound memory model that can successfully address the open problems.

- **Obj-2**: Propose and study architecture features to meet the performance and energy efficiency to realize Obj-I.

- **Obj-3**: Verify our solution through a qualitative/quantitative on credible platforms and benchmark programs.
Toward A Dynamic Multithreaded Execution Model and Abstract Machine Based on Codelets

Global Distributed Shared Address Space

Tree of memory “trunks”/blocks

“commit-once semantics” + LC model

Features of Codelet Machine: 1) distributed garbage collection, (2) percolation, (3) cache under write-once semantic, (4) hardware scheduling support, (5) caching of SSA events, etc.
An Example Platform – The SWARM (ETI) and DART (CAPSL)
Open-Questions-on-Memoty-Models
SWARM Execution Model
Conclusion

• Mr. X’s remark - a wake-up call to all of us in this room !!

• We must continue our R&D to give a fresh look at these open problems along the classical path

• A *fresh look* is needed on these open problems!
Acknowledgements

• Our Sponsors
• Intel UHPC Team
• CAPSL Team (Juergen, Kelly, Daniel, Elkin, Stéphane, Haitao, Robert, Josh S, Josh L, Aaron, etc.)
• ETI Team Other Collaborators (Dennis, Sterling, Sarkar, Inte TG team, PNNL team,
• My Host
CAPSL – The Team

Joshua D. Suetterlein
Kelly Livingston
Aaron M. Landwehr
Elkin Garcia
Chen Chen