Macroservers: A New High-Level Programming and Execution Model for PIM-Based Scalable Architectures

Hans P. Zima

Engineering and Communications Infrastructure Section (366)

zima@jpl.nasa.gov

JPL IT Symposium November 2002
OUTLINE

• 1 Introduction
• 2 Processor-in-Memory (PiM) Systems
• 3 Macroservers
• 4 Research Issues
• 5 Conclusion
The Gap Between Processor and DRAM Performance


JPL IT Symposium November 2002
Processor-in-Memory

- Integration of CMOS logic/DRAM memory
- Replication of PIM nodes across module
- Multithreaded processors
- Huge improvement of on-chip bandwidth
- Lightweight threads in memory
- Lightweight communication via parcels
- Efficient memory operations
- No data caches

PIM module (chip)
PIM-Based System Configurations

• *PIM as a special-purpose processor*: Mitsubishi M32 R/D (1996), IRAM (1997): embedded systems on a chip with vector processing logic

• *PIM as memory in a conventional system*: FlexRAM (UIUC, 1997), DIVA (University of Notre Dame, USC/ISI, Caltech, since 1998)

• *PIM as a part of a memory hierarchy*: HTMT (1998-), Cascade (2002-)

• *PIM array*: Blue Gene (IBM, since 1999), PIM-Lite (Notre Dame, 2002), Gilgamesh (Caltech/JPL)

JPL IT Symposium November 2002
IBM Blue Gene BG/C

Five Steps to a Petaflop Computer

Blue Gene

Tower

Board

Chip

Processor

1 Petaflop

16 Teraflops

2 Teraflops

32 Gigaflops

1 Gigaflop
Application Domains

- Large-scale ground-based numerical computation
- Spaceborne systems (including autonomy)
- Monitoring and control (real-time)

Strategy

- Highly replicated fine-grain processors
- Virtual paged memory management
- Message-driven execution
- Multithreading
- Support for irregular data structures
A Software Architecture for PIM-Based Systems

Very high-level language

High-level language

Libraries

MACROSERVER SPACE

JPL IT Symposium November 2002
Macroservers

- Middleware for object-based runtime management of data and threads; target for high-level languages or very high-level specification systems

- Distributed collections are a key concept: they consist of a data structure, together with a specification of a distribution (across "processors").

- Specialization: intrinsic object classes, distributions, and compilation methods (e.g., for sparse matrices)

- Application of special operations to distributed collections or sets of collections: standard operators, reductions, prefix operations, filter

- Futures
  - spawning of structured data parallelism
  - creation of skeletons for the coordination of (data parallel or heterogeneous) tasks [Opus]
  - arbitrarily recursive task structures

JPL IT Symposium November 2002
Column-block distribution of a 2D-matrix

Regular distributions such as this can be easily handled in the compiler/runtime system.
Example: Distribution of a Tree Structure
Example: A Multiblock Grid Collection

- define partition of "processor" set
- distribute grids
- process grids in parallel
- run individual solvers in parallel

Source: C.B. Allen, Bristol, UK
Example: MRD/CRS Sparse Matrix Distribution
Requirements for Irregular and Dynamic Applications

- **General data structures**

- **General methods for distributing and aligning data**
  (regular distributions may not reflect locality in physical space)

- **General mechanisms for data/thread affinity**
  (allow a dynamic mapping of thread groups to memory segments associated with a data partition)

- **Dynamic manipulation of data distributions, alignments, and affinity must be efficient**
  (apart from adaptive problems such as SAMR dynamic redistributions are even needed for regular problems such as ADI)
**Distribution and Alignment Control**

*Data distribution:*
- mapping data to "processors"
- standard or user-defined
- first-class distribution objects
- dynamic redistribution

*Data alignment:*
- establishing mapping between data domains

*Work distribution:*
- mapping sets of threads to "processors"

*Work alignment:*
- mapping sets of threads to distribution segments

[Diagram showing data distribution and alignment]
Intelligent Software

- Self-adapting software

- Automatic, application-guided data and work distribution

- Feedback-oriented compilation: interaction of compiler with dynamic performance analysis subsystem

- Verification and validation support

- Fault tolerance support

JPL IT Symposium  November 2002
Conclusion

- **Macroservers**: middleware for object-based runtime management of data and threads in homogeneous PIM-based architectures

- **Target for high-level languages or very high-level specification systems**

- **Full control of locality and parallelism**

- **Mapped to local node-servers on a PIM array**

JPL IT Symposium  November 2002