OPERA Software Architecture

OSA

Steve Crago

Janice McMahon

USC/ISI-East

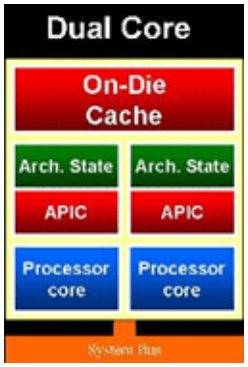
May 29, 2008

Outline

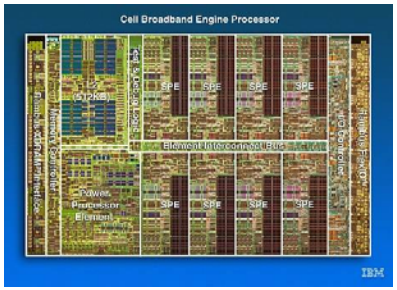
- **Introduction**
- **OPERA Software Environment**
- **Software Fault Tolerance for OPERA**

Multicore Trends

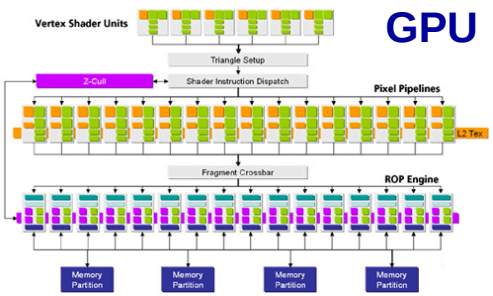
- Power density and complexity problems have driven current and next generation processors to have multiple cores on chip
  - Intel Core Duo
  - AMD
  - 8 core SUN Niagara-2
  - Tilera TILE64
- On-chip parallelism improves throughput of multiple applications, but results in programming challenges
  - Single applications must be parallelized
  - Parallel applications must be scalable
  - Requires highly skilled programmers or *better tools*



x86 Multi-Core

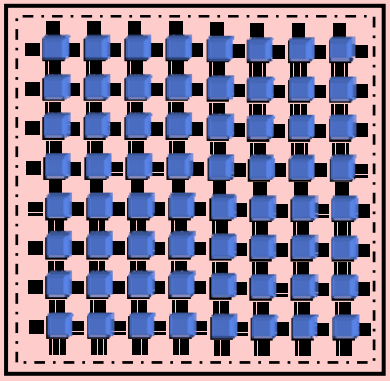


Cell



GPU

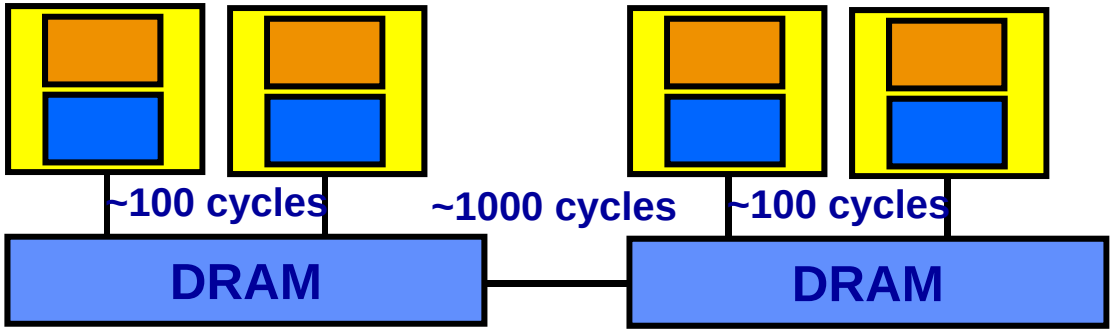
TILE64



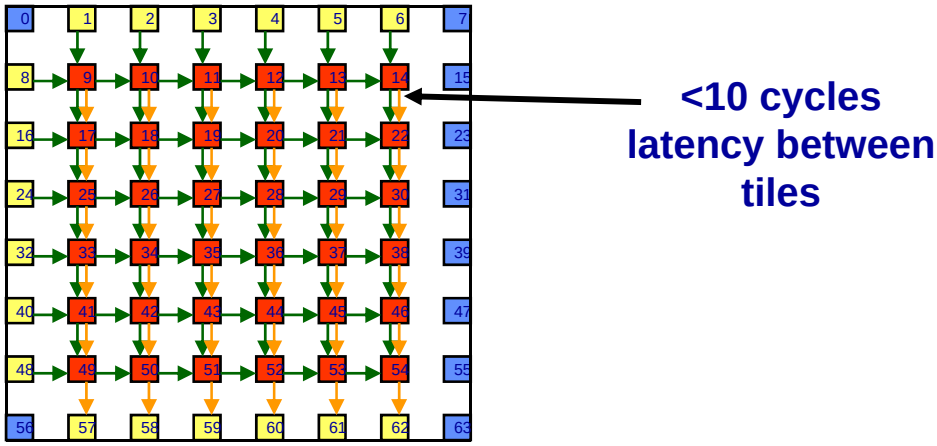
Interprocessor Communication

- **Low latency connections between compute tiles expose new software issues (multi-core ≠ Symmetric Multiprocessor ≠ High Performance Computer)**

Traditional Multiprocessor



Multi-Core Architecture



Performance Sources

- **Instruction Level Parallelism**
  - Fine grained
  - Available in general-purpose legacy code
  - Exploited by traditional superscalar and VLIW uniprocessors
  - Typically limited by branch prediction and dependencies
- **Data Level Parallelism**
  - Efficient to exploit
  - Available in streaming applications
  - Relatively easy for programmer to specify
  - Exploited by multimedia extensions, Cell, SIMD architectures
- **Thread Level Parallelism**
  - Executes tasks in parallel
  - Needed for scalability
  - Trickier to program or extract from program
  - Initial focus of commercial multi-core architectures

Multi-core tools must consider these sources of parallelism simultaneously. This differentiates multi-core tools from traditional uniprocessor and multiprocessor tools.

Government/Industry Multi-core Roles

- **Industry**
  - Hardware: stamp x86 cores on a chip (e.g. Intel, AMD)
  - Hardware: develop aggressive multi-core chip for specific commercial markets (e.g. Tilera)
  - Evolutionary software tools and programming support
  - General-purpose software research and education (e.g. fund universities to teach students how to program multi-core)
- **Government Responsibilities**
  - Special needs (e.g. radiation hardening)
  - Application libraries (e.g. MPI, VSIPL)
  - Domain-specific tools
  - Run-time management for dynamic scenarios and autonomous operation (resource management and fault tolerance)
  - Longer term research

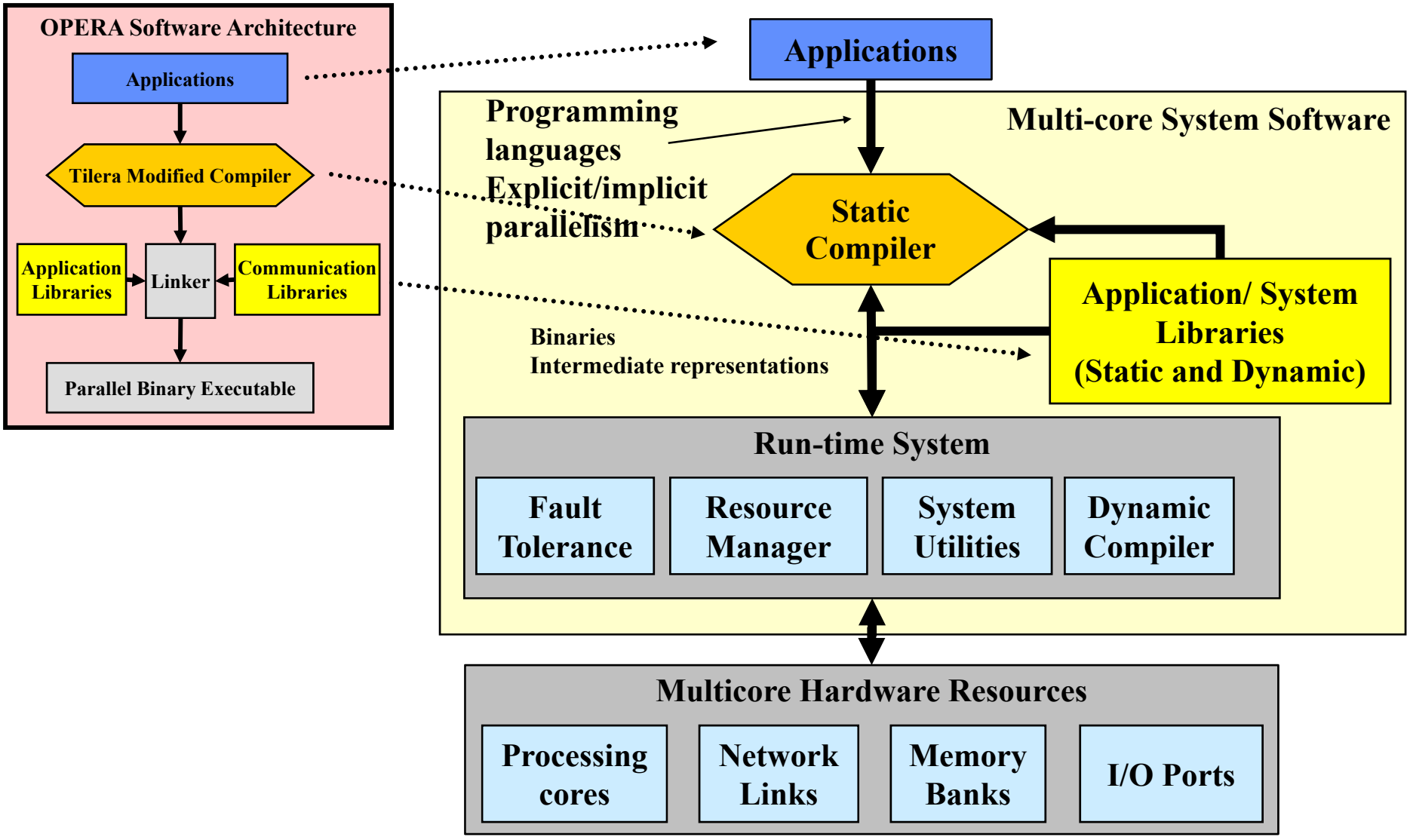
Outline

- Introduction
- OPERA Software Environment
- Software Fault Tolerance for OPERA

OPERA Software Architecture Goals

- **Enable application demonstration of OPERA processor**
  - Provide basic functionality to allow hardware technology demonstration
- **Leverage commercial Tilera software base**
  - Current support is for sequential compiler and library for message passing and shared memory
- **Provide baseline programming environment for future missions**
  - Based on familiar programming models
  - Standard-compliant APIs
- **Provide technology base for future software technologies**
  - Domain-specific parallelization
  - Dynamic resource management
  - Software fault tolerance

Long-Term Multicore System Software Vision



OSA Funded Tasks

- **Tool extensions**
  - **Floating point extensions: compiler, simulator, debugger, iLib library**
  - **Legacy software support: C++, MPI, OpenMP, VSIPL, pVSIPL++, OpenPNL**
- **Performance and productivity tools**
  - **Parallel performance analysis**
  - **Parallel debug**
  - **Run-time monitor and run-time system**
  - **Fine-grain parallel compiler**
- **Applications**
  - **JPEG2K**
  - **CAF**
  - **Co-add**
  - **On-board processing**

Outline

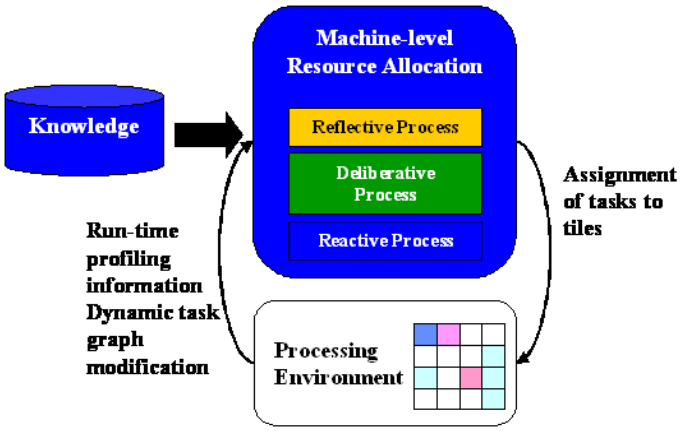
- Introduction
- OPERA Software Environment
- Software Fault Tolerance for OPERA

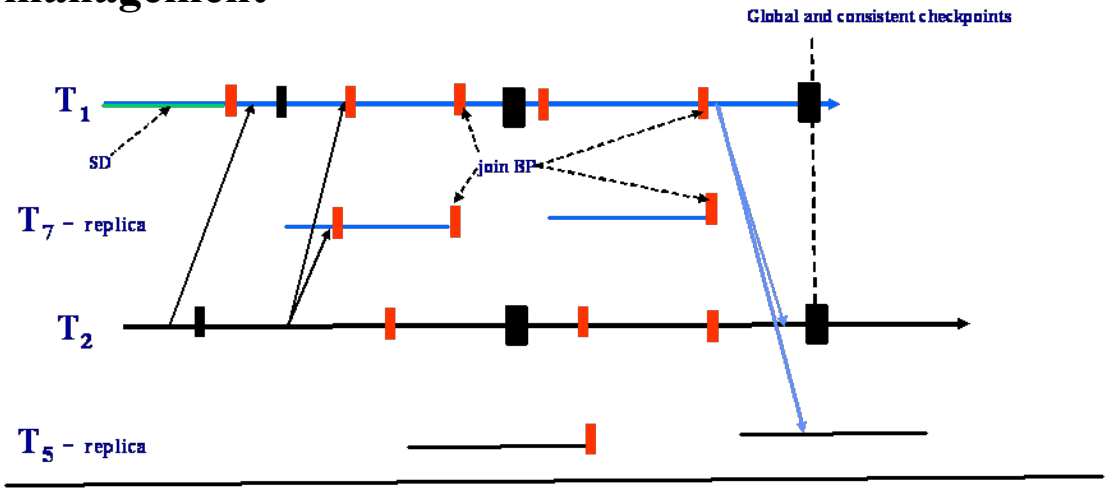
Software Fault Tolerance

- Rad-hard by design (RHBD) as applied to OPERA will provide protection for total dose, latchup, and single-event upsets
  - Rate of updates can be traded off with cost of protection
  - RHBD can be used in combination with system-level software techniques
- Other sources of faults will still exist, e.g.
  - Software error
  - Physical damage
- Software fault tolerance provides mitigation for faults while minimizing overhead and can work in conjunction with RHBD to provide overall mission reliability

Run-time Research

- Resource management and introspection
  - Core allocation
  - Power management
  - Introspection hooks
    - Performance monitoring
    - Behavior monitoring
    - Support for programming and performance tuning
- Fault tolerance
  - Support from compiler and run-time system
  - Limited redundancy
  - Check-pointing and roll-back
  - Interaction with resource management





Fault Tolerance Opportunities

- **Redundancy available**
  - **Cores**
  - **Networks**
  - **Memory interfaces**
  - **I/O**
- **Programmability**
  - **Fault tolerance and performance can be tuned according to application needs**

Fault Tolerance Challenges

- **Increased state**
  - **Cores**
  - **Networks**
  - **Memory interfaces**
  - **I/O**
- **Programmability**
  - **Programming *can* be topology-dependent and tied to physical location**

Summary

- OPERA is an opportunity to provide unprecedented general-purpose performance for space
- OPERA software will build on commercial software to provide an environment suitable for government applications
- OPERA provides both challenges and opportunities for fault tolerance
  - Being addressed through both hardware and software