CPEG 852 — Advanced Topics in Computing Systems
The EARTH Program Execution Model
Hybrid von Neumann/Dataflow Models

Stéphane ZUCKERMAN

Computer Architecture & Parallel Systems Laboratory
Electrical & Computer Engineering Dept.
University of Delaware
140 Evans Hall Newark,DE 19716, United States
szuckerm@udel.edu

October 20-27, 2015
Outline

1. Introduction
   - Secret Origins of EARTH

2. The EARTH Program Execution Model

3. The EARTH Abstract Machine Model
   - The EARTH Abstract Machine

4. EARTH-Manna: An Implementation of the EARTH Architecture Model

5. Programming Models for Multithreaded Architectures
   - Features of Multithreaded Programming Models
   - EARTH Instruction Set
   - The EARTH Benchmark Suite (EBS)
   - Programming Examples
   - Compilation Environment, Revisited

6. Summary
# Outline

1. **Introduction**
   - Secret Origins of EARTH

2. The EARTH Program Execution Model

3. The EARTH Abstract Machine Model
   - The EARTH Abstract Machine

4. EARTH-Manna: An Implementation of the EARTH Architecture Model

5. Programming Models for Multithreaded Architectures
   - Features of Multithreaded Programming Models
   - EARTH Instruction Set
   - The EARTH Benchmark Suite (EBS)
   - Programming Examples
   - Compilation Environment, Revisited

6. Summary
Outline

1. **Introduction**
   - Secret Origins of EARTH

2. **The EARTH Program Execution Model**

3. **The EARTH Abstract Machine Model**
   - The EARTH Abstract Machine

4. **EARTH-Manna: An Implementation of the EARTH Architecture Model**

5. **Programming Models for Multithreaded Architectures**
   - Features of Multithreaded Programming Models
   - EARTH Instruction Set
   - The EARTH Benchmark Suite (EBS)
   - Programming Examples
   - Compilation Environment, Revisited

6. **Summary**
At the beginning of the 1990’s, dataflow machines are considered obsolete compared to their vector and superscalar cousins.

Some people still think dataflow as a founding principle for models of computation is still sound.

Idea: mix RISC-like processors for the low-level execution and dataflow semantics for high-level concepts.

Technically, this is more or less the definition of macro-dataflow.

EARTH is a bit different from macro-dataflow – see next slide.
Macro-Dataflow: a Description

- Concept: take a sequence of instructions, and group them into a macro-dataflow actor
- A macro-dataflow actor has input arcs (for the “tokens”) and output arcs
- A macro-dataflow actor has no state except the one formed by its input tokens
- Firing rule: when all input arcs have a token present, the macro-dataflow actor may fire
- A macro-dataflow actor always places tokens on its outputs arcs all at once
Why EARTH Does Not Quite Follow a Macro-Dataflow Model

- In EARTH, actors do execute an uninterruptible sequence of instructions (like in macro-dataflow)
- But an EARTH actor can signal/produce data at any time during its execution
  - It is more “loose” in its asynchrony
What Is a Thread in EARTH?

- Parallel function invocation (threaded function invocation)
- A code sequence defined (by the user or a compiler) to be a thread (fiber)
- Usually, a threaded function’s body will be partitioned into multiple threads/fibers
Outline

1 Introduction
   - Secret Origins of EARTH

2 The EARTH Program Execution Model

3 The EARTH Abstract Machine Model
   - The EARTH Abstract Machine

4 EARTH-Manna: An Implementation of the EARTH Architecture Model

5 Programming Models for Multithreaded Architectures
   - Features of Multithreaded Programming Models
   - EARTH Instruction Set
   - The EARTH Benchmark Suite (EBS)
   - Programming Examples
   - Compilation Environment, Revisited

6 Summary
A Simple Example: Naïve Fibonacci

```c
u64 fib(u64 n) {
    if (n < 2) return n;
    return fib(n-1) + fib(n-2);
}
```
A Simple Example: Naïve Fibonacci

Computation Tree Example

\[
\begin{align*}
\text{fib}(4) & \quad \text{fib}(3) \\
& \quad \text{fib}(2) \quad \text{fib}(1) \\
& \quad \text{fib}(1) \quad \text{fib}(0) \\
& \quad \text{fib}(1) \quad \text{fib}(0) \\
& + \quad \text{fib}(2) \quad \text{fib}(0)
\end{align*}
\]
A Simple Example: Naïve Fibonacci Activation Frame Tree

```
fib(n-2)  fib(n-3)  
\downarrow     \downarrow
fib(n-1)  fib(n-2)  
\downarrow     \downarrow
fib(n) 
```

 SYNC Slot
 Local Vars
 Caller's <fp, ip>

Links between frames
Known Short Latencies, Known Long Latencies, and Unknown Latencies

An Example

```c
int f(int *x, int i, int j) {
    int a, b, sum, prod, fact;
    int r1, r2, r3;
    a = x[i];
    fact = 1;
    fact = fact * a;
    b = x[j];
    sum = a + b;
    prod = a * b;
    r1 = g(sum);
    r2 = g(prod);
    r3 = g(fact);
    return r1 + r2 + r3;
}
```
Known Short Latencies, Known Long Latencies, and Unknown Latencies
Partitioning Into Fibers

```plaintext
a = x[i];
fact = 1;

fact = fact * a;
b = x[j];

sum = a + b;
prod = a * b;
r1 = g(sum);
r2 = g(prod);
r3 = g(fact);

return r1 + r2 + r3;
```
Fibers

- A fiber shares its enclosing “frame” with other fibers within the same threaded function invocation.
- The state of a fiber includes:
  - Its instruction pointer
  - Its “temporary register set”
- A fiber is “ultra-lightweight:” it does not need dynamic storage (frame) allocation.
- Our focus: **non-preemptive** threads – which we call **fibers**
The EARTH Execution Model
Fibers Firing Rule

- A fiber becomes enabled if it has received all input signals.
- An enabled fiber may be selected for execution when the required hardware resource has been allocated.
- When a fiber finishes its execution, a signal is sent to all destination threads to update the corresponding synchronization slots.
Fibers States

Thread Created / Thread Terminated

Dormant

Synchronization Received

Enabled

Thread Completed

CPU Ready

Active
The EARTH Model of Computation
Two levels of multithreading:
- Threaded procedures
- Fibers
EARTH vs. Cilk

Figure: EARTH Model

Figure: Cilk Model
The Fiber Execution Model
The Fiber Execution Model
The Fiber Execution Model
The Fiber Execution Model
The Fiber Execution Model

- Signal token
- Arrived signals
- Total signals

Diagram showing the interaction and flow of signals and tokens in the Fiber Execution Model.
The Fiber Execution Model
The Fiber Execution Model
The Fiber Execution Model
The Fiber Execution Model
The Fiber Execution Model

- Signal token
- Arrived signals
- Total signals
The Fiber Execution Model
The Fiber Execution Model

Signal token  Arrived signals  Total signals

1 1
2 2
4 4
Outline

1 Introduction
   - Secret Origins of EARTH

2 The EARTH Program Execution Model

3 The EARTH Abstract Machine Model
   - The EARTH Abstract Machine

4 EARTH-Manna: An Implementation of the EARTH Architecture Model

5 Programming Models for Multithreaded Architectures
   - Features of Multithreaded Programming Models
   - EARTH Instruction Set
   - The EARTH Benchmark Suite (EBS)
   - Programming Examples
   - Compilation Environment, Revisited

6 Summary
Outline

1 Introduction
   • Secret Origins of EARTH

2 The EARTH Program Execution Model

3 The EARTH Abstract Machine Model
   • The EARTH Abstract Machine

4 EARTH-Manna: An Implementation of the EARTH Architecture Model

5 Programming Models for Multithreaded Architectures
   • Features of Multithreaded Programming Models
   • EARTH Instruction Set
   • The EARTH Benchmark Suite (EBS)
   • Programming Examples
   • Compilation Environment, Revisited

6 Summary
The EARTH Abstract Machine
The EARTH Abstract Machine

[Diagram showing the EARTH Abstract Machine with memory bus, PE (Processing Element) blocks, LOCAL MEMORY, EU (Execution Unit), RQ (Request Queue), EQ (Execution Queue), SU (Storage Unit), and node connections through an interconnection network.]
EARTH’s Evaluation Platforms

EARTH-Manna
Implement EARTH on a bare metal tightly coupled multi-processor
EARTH’s Evaluation Platforms

**EARTH-Manna**
Implement EARTH on a bare metal tightly coupled multi-processor

**EARTH-IBM-SP**
Plan to implement EARTH on an off-the-shelf commercial parallel machine (IBM SP2/SP3)
EARTH’s Evaluation Platforms

**EARTH-Manna**
Implement EARTH on a bare metal tightly coupled multi-processor

**EARTH-IBM-SP**
Plan to implement EARTH on an off-the-shelf commercial parallel machine (IBM SP2/SP3)

**EARTH on Clusters**
- EARTH on Beowulf
- Implement EARTH on a cluster of UltraSPARC SMP workstations connected by Fast Ethernet
### EARTH’s Evaluation Platforms

<table>
<thead>
<tr>
<th>Platform</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>EARTH-Manna</strong></td>
<td>Implement EARTH on a <strong>bare metal</strong> tightly coupled multi-processor</td>
</tr>
<tr>
<td><strong>EARTH on Clusters</strong></td>
<td>- EARTH on Beowulf</td>
</tr>
<tr>
<td></td>
<td>- Implement EARTH on a cluster of UltraSPARC SMP workstations connected by Fast Ethernet</td>
</tr>
<tr>
<td><strong>EARTH-IBM-SP</strong></td>
<td>Plan to implement EARTH on an off-the-shelf commercial parallel machine (IBM SP2/SP3)</td>
</tr>
</tbody>
</table>

Note—Benchmark codes were all written using EARTH Threaded-C: The API for EARTH Execution and Abstract Machine Models
Outline

1. Introduction
   - Secret Origins of EARTH

2. The EARTH Program Execution Model

3. The EARTH Abstract Machine Model
   - The EARTH Abstract Machine

4. EARTH-Manna: An Implementation of the EARTH Architecture Model

5. Programming Models for Multithreaded Architectures
   - Features of Multithreaded Programming Models
   - EARTH Instruction Set
   - The EARTH Benchmark Suite (EBS)
   - Programming Examples
   - Compilation Environment, Revisited

6. Summary
Open Issues

- Can a multithreaded program execution model support high scalability for large-scale parallel computing while maintaining high processing efficiency?
- If so, can this be achieved without exotic hardware support?
- Can these open issues be addressed both qualitatively and quantitatively with performance studies of real-life benchmarks?
The EARTH-MANNA Multiprocessor Testbed

Cluster → Cluster → Cluster → Cluster

Cross-Bar Hierarchies

Cluster → Cluster → Cluster → Cluster

Node → Node → Node → Node

Crossbar

Node → Node → Node → Node

32MiB DRAM

Node
Main Features of EARTH Multiprocessor

- Fast thread context switching
- Efficient parallel function invocation
- Good support of fine-grain dynamic load-balancing
- Efficient support of split-phase transactions
- Brings together the concept of fibers and dataflow
Main Features of EARTH Multiprocessor

- Fast thread context switching
- Efficient parallel function invocation
- Good support of fine-grain dynamic load-balancing
- Efficient support of split-phase transactions
- Brings together the concept of fibers and dataflow
- All that using off-the-shelf microprocessors!
Figure: EARTH Compilation Environment

Figure: EARTH-C Compiler Environment
Performance Study of EARTH

- Overview
- Microbenchmarking:
  - Stress-testing
  - Measure performance of basic EARTH mechanisms for communication and synchronization
- Kernel benchmarking:
  - Speedup
  - USE value (Uni-node Support Efficiency)
    - i.e., compare pure sequential kernel vs. single-node EARTH kernel
  - Latency tolerance capacity

Note—it is important to define your own performance “features” and/or “parameters” that best distinguishes your model from your competitors.
## EARTH Benchmark Suite (EBS)

<table>
<thead>
<tr>
<th>Benchmark Name</th>
<th>Problem Size</th>
<th>Problem Domain</th>
<th>Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ray Tracing</td>
<td>512 × 512</td>
<td>Image Processing</td>
<td>Class A</td>
</tr>
<tr>
<td>Wave-2D</td>
<td>150 × 150</td>
<td>Fluid Dynamics</td>
<td>Class A</td>
</tr>
<tr>
<td><strong>Tomcatv</strong></td>
<td>257</td>
<td>Scientific computation</td>
<td>Class A</td>
</tr>
<tr>
<td>2D-SLT</td>
<td>80 × 80</td>
<td>Fluid Dynamics</td>
<td>Class A</td>
</tr>
<tr>
<td><strong>Matrix Multiply</strong></td>
<td>480 × 480</td>
<td>Numerical computation</td>
<td>Class A</td>
</tr>
<tr>
<td>Barnes-Hut</td>
<td>8192 bodies</td>
<td>N-Body simulation</td>
<td>Class B</td>
</tr>
<tr>
<td>MP3D</td>
<td>18 K particles</td>
<td>Fluid Flow simulation</td>
<td>Class B</td>
</tr>
<tr>
<td>EM3D</td>
<td>20 K nodes</td>
<td>Electromagnetic wave simulation</td>
<td>Class B</td>
</tr>
<tr>
<td>Sampling Sorting</td>
<td>64 K</td>
<td>Sorting problem</td>
<td>Class B</td>
</tr>
<tr>
<td>Gauss Elimination</td>
<td>720 × 720</td>
<td>Numerical computation</td>
<td>Class B</td>
</tr>
<tr>
<td>Protein Folding</td>
<td>3 × 3 × 3 cube</td>
<td>Chemistry</td>
<td>Class B</td>
</tr>
<tr>
<td>Eigenvalue</td>
<td>2999</td>
<td>Numerical computation</td>
<td>Class B</td>
</tr>
<tr>
<td>Vertex Enumeration</td>
<td>10</td>
<td>Pivot-based searching</td>
<td>Class B</td>
</tr>
<tr>
<td>TSP</td>
<td>10</td>
<td>Graph searching</td>
<td>Class B</td>
</tr>
<tr>
<td>Paraffins</td>
<td>20</td>
<td>Chemistry</td>
<td>Class B</td>
</tr>
<tr>
<td>N-Queen</td>
<td>12</td>
<td>Graph searching</td>
<td>Class B</td>
</tr>
<tr>
<td>Power</td>
<td>10000</td>
<td>Power system optimization</td>
<td>Class B</td>
</tr>
<tr>
<td>Voronoi</td>
<td>64 K</td>
<td>Graph Partitioning</td>
<td>Class B</td>
</tr>
<tr>
<td>Heuristic-TSP</td>
<td>32 K</td>
<td>Searching problem</td>
<td>Class B</td>
</tr>
<tr>
<td>Tree-Add</td>
<td>1 M</td>
<td>Graph Searching</td>
<td>Class B</td>
</tr>
</tbody>
</table>

Highlighted benchmarks are implemented using a portable version of Threaded-C
Efficient multithreading support is possible with off-the-shelf processor nodes with low overhead

- At the time: context-switch time $\approx 35$ cycles
- Nowadays, this figure would most likely be bigger by at least one order of magnitude, probably $\approx 20$ times bigger
  - This is speculation, but even with a bare-metal implementation, there is a $\approx 3 \times$ difference between memory bus and processor frequencies.

A multithreaded program execution model can make a big difference

- Results from the EARTH Benchmark Suite (EBS)
1. Introduction
   - Secret Origins of EARTH

2. The EARTH Program Execution Model

3. The EARTH Abstract Machine Model
   - The EARTH Abstract Machine

4. EARTH-Manna: An Implementation of the EARTH Architecture Model

5. Programming Models for Multithreaded Architectures
   - Features of Multithreaded Programming Models
   - EARTH Instruction Set
   - The EARTH Benchmark Suite (EBS)
   - Programming Examples
   - Compilation Environment, Revisited

6. Summary
Outline

1. Introduction
   - Secret Origins of EARTH

2. The EARTH Program Execution Model

3. The EARTH Abstract Machine Model
   - The EARTH Abstract Machine

4. EARTH-Manna: An Implementation of the EARTH Architecture Model

5. Programming Models for Multithreaded Architectures
   - Features of Multithreaded Programming Models
   - EARTH Instruction Set
   - The EARTH Benchmark Suite (EBS)
   - Programming Examples
   - Compilation Environment, Revisited

6. Summary
Threaded-C: A Base Language

- Serves as a target language for high-level language compilers
- Serves as a machine language for the EARTH architecture
The Role of Threaded-C

C

High-Level Language Translation

Fortran

Users

Threaded-C

Threaded-C Compiler

EARTH Platforms
Outline

1. Introduction
   - Secret Origins of EARTH

2. The EARTH Program Execution Model

3. The EARTH Abstract Machine Model
   - The EARTH Abstract Machine

4. EARTH-Manna: An Implementation of the EARTH Architecture Model

5. Programming Models for Multithreaded Architectures
   - Features of Multithreaded Programming Models
   - EARTH Instruction Set
   - The EARTH Benchmark Suite (EBS)
   - Programming Examples
   - Compilation Environment, Revisited

6. Summary
Features of Threaded Programs

- Thread partition
  - Thread length vs. useful parallelism
  - Where to “cut” a dependence and create a “split-phase?”
- Split-phase synchronization and communication
- Parallel threaded function invocation
- Dynamic load-balancing
- Other advanced features: fibers and dataflow
The EARTH Operation Set

### Base operations

- Thread synchronization and scheduling ops
  - SPAWN
  - SYNC

- Split-phase data & synchronization operations
  - GET_SYNC
  - DATA_SYNC

- Threaded function invocation and load-balancing operations
  - INVOKE
  - TOKEN
Basic instructions:

- Arithmetic, logic, branching
- Typical RISC instructions, e.g., those from the i860

Thread Switching

- FETCH_NEXT

Synchronization

- SPAWN fp, ip
- SYNC fp, ss_off
- INIT_SYNC ss_off, sync_cnt, reset_cnt, ip
- INCR_SYNC fp, ss_off, value

Data transfers & synchronization

- DATA_SPAWN value, dest_addr, fp, ip
- DATA_SYNC value, dest_addr, fp, ss_off
- BLOCKDATA_SPAWN src_addr, dst_addr, size, fp, ip
- BLOCKDATA_SYNC src_addr, dst_addr, size, fp, ss_off
Split-phase data requests

- GET_SPAWN src_addr, dst_addr, fp, ip
- GET_SYNC src_addr, dst_addr, fp, ss_off
- GET_BLOCK_SPAWN src_addr, dst_addr, fp, ip
- GET_BLOCK_SYNC src_addr, dst_addr, fp, ss_off

Function invocation

- INVOKE dst_PE, func_name, num_params, params
- TOKEN func_name, num_params, params
- END_FUNCTION
Outline

1. Introduction
   - Secret Origins of EARTH

2. The EARTH Program Execution Model

3. The EARTH Abstract Machine Model
   - The EARTH Abstract Machine

4. EARTH-Manna: An Implementation of the EARTH Architecture Model

5. Programming Models for Multithreaded Architectures
   - Features of Multithreaded Programming Models
   - EARTH Instruction Set
   - The EARTH Benchmark Suite (EBS)
   - Programming Examples
   - Compilation Environment, Revisited

6. Summary
- **Ray Tracing** is a program for rendering 3-D photo-realistic images
- **Protein Folding** is an application that computes all possible foldings structures of a given polymer
- **TSP** is an application to find a minimal-length Hamiltonian cycle in a graph with N cities and weighted paths.
- **Tomcatv** is one of the SPEC benchmarks which operates upon a mesh
- **Parrafin**s is another application which enumerates distinct isomers paraffins
- **2D-SLT** is a program implementing the 2D-SLT Semi-Lagrangian Advection Model on a Gaussian Grid for numerical weather predication
- **N-Queens** is a benchmark program typical of graph searching problem.
Outline

1 Introduction
   - Secret Origins of EARTH

2 The EARTH Program Execution Model

3 The EARTH Abstract Machine Model
   - The EARTH Abstract Machine

4 EARTH-Manna: An Implementation of the EARTH Architecture Model

5 Programming Models for Multithreaded Architectures
   - Features of Multithreaded Programming Models
   - EARTH Instruction Set
   - The EARTH Benchmark Suite (EBS)
   - Programming Examples
   - Compilation Environment, Revisited

6 Summary
Reminder: Tree of Activation Frames

fib(n-2) → fib(n-1) → fib(n) → fib(n-2)

SYNC Slot
Local Vars
Caller's <fp, ip>

Links between frames
The Fibonacci Example

### Definition: Fibonacci Sequence

\[
\begin{align*}
  U_0 &= 0 \\
  U_1 &= 1 \\
  U_n &= U_{n-1} + U_{n-2}, \quad \forall n \in \mathbb{N}
\end{align*}
\]

```c
u64 fib(u64 n) {
    if (n < 2) return n;
    return fib(n-1) + fib(n-2);
}
```
The Fibonacci Example
Threaded-C Version

```c
THREADED fib(u64 n, u64* result, sslot_t done) {
    THREAD -0:
        u64 sum1 = 0, sum2 = 1;
        sslot_t slot1 = ... , /*, (0,0), */,
            slot2 = ... , /*, (2,2), */;
        if (n < 2) {
            DATA_RSYNC(n, result, done);
        } else {
            TOKEN (fib, n-1, &sum1, slot1);
            TOKEN (fib, n-2, &sum2, slot2);
        }
        END_THREAD();
    THREAD -1:
        DATA_RSYNC(sum1+sum2;, result, done);
        END_THREAD();
    END_FUNCTION;
}
```
General Matrix Multiplication (GEMM)

Definition

Although the definition is general, we set our numbers in either $\mathbb{R}$ or $\mathbb{C}$. We generalize with the set $\mathbb{K}$.

Let $A$, $B$, and $C$ be matrices, with $A_{M,K}$, $B_{K,N}$, $C_{M,N} \in \mathbb{K} \times \mathbb{K}$; let $\alpha, \beta \in \mathbb{K}$. Then,

$$C_{M,N} \leftarrow \beta \times C_{M,N} + \alpha \times A_{M,K} \times B_{K,N}$$

One element $c_{i,j}$ of $C_{M,N}$ is computed as such:

$$c_{i,j} = \beta \cdot c_{i,j} + \alpha \cdot \sum_{k=1}^{k=K} a_{i,k} \cdot b_{k,j}, \quad a_{i,k} \in A_{M,K}, \quad b_{k,j} \in B_{K,N}$$
Note—To simplify the problem, we assume $\beta = 0$, $\alpha = 1, 0$. In other words, we compute $C_{N,N} = A_{N,N} \times B_{N,N}$.

```c
void dgemm(const double beta, double* C,
            const double alpha,
            const double* A, const double* B,
            const size_t N)
{
    for (size_t i = 0; i < N; ++i) {
        for (size_t j = 0; j < N; ++j) {
            c[i*N+j] = beta * c[i*N+j];
            for (size_t k = 0; k < N; ++k)
                c[i*N+j] += alpha * A[i*N+k] * b[k*N+j];
        }
    }
}
```
The Fibonacci Example

Threaded-C Version — Outer Product

Note—We assume the B matrix is correctly transposed to allow row-major traversal.

```c
THREADED void dgemm(double* C, const double* A, const double* B,
                       const size_t N)
{
    double *row_a, *column_b;
    sslot_t slot0 = init_sslot(0,0),
                 slot1 = init_sslot(N*N,N*N);

    THREAD -0:
    for (size_t i = 0; i < M; ++i) {
        for (size_t j = 0; j < N; ++j) {
            row_a = a[i];
            column_b = b[j];
            for (size_t k = 0; k < K; ++k)
                TOKEN (inner, &c[i*N+j], row_a, column_b, slot2);
        }
    }
    END_THREAD();

    THREAD -1:
    RETURN(); //, do, nothing
    END_THREAD();

    END_FUNCTION;
}
```
The Fibonacci Example
Threaded-C Version — Inner Product

Note—We assume the B matrix is correctly transposed to allow row-major traversal.

```c
THREADED void inner(double* result, const size_t N,
                      const double* A, const double* B,
                      sslot_t done)
{
    double sum, *row_a, *column_b;
    sslot_t slot0 = init_sslot(0,0),
              slot1 = init_sslot(2,2);

    THREAD-0:
        BLKMOV_SYNC(A, row_a, N, slot1);
        BLKMOV_SYNC(B, column_b, N, slot1);
        sum = 0.0;
        END_THREAD();

    THREAD-1:
        for (size_t i = 0; i < N; ++i)
            sum += row_a[i] * column_b[j];

        DATA_RSYNC(*result += sum;, done);

        END_THREAD();

    END_FUNCTION;
```
1 Introduction
   - Secret Origins of EARTH

2 The EARTH Program Execution Model

3 The EARTH Abstract Machine Model
   - The EARTH Abstract Machine

4 EARTH-Manna: An Implementation of the EARTH Architecture Model

5 Programming Models for Multithreaded Architectures
   - Features of Multithreaded Programming Models
   - EARTH Instruction Set
   - The EARTH Benchmark Suite (EBS)
   - Programming Examples
   - Compilation Environment, Revisited

6 Summary
Compilation Environment, Revisited

C

EARTH-C

McCAT

Program Dependence Analysis

EARTH SIMPLE

EARTH-SIMPLE

Split-Phase Analysis

Build Data-Dependence Graph

Compute Remote Level

Merge Statements

Fiber Synchronization

Fiber Scheduling

Fiber Code Generation

Thread/Fiber Partitioning

Threaded-C

EARTH-C Compiler

Thread/Fiber Generation

EARTH Compilation Environment
Compilation Environment, Revisited

EARTH-C

- Simplify go-to elimination
- Local function inlining
- Points-to analysis
- Heap analysis
- R/W set analysis
- Array dependence tester

EARTH-SIMPLE

- forall loop detection
- Loop partitioning

EARTH-SIMPLE

- Build hierarchical DDG
- Thread generation

Code Generation

Phase I
- Standard McCat
- Analyses & Transformations

Phase II
- Parallelization

Phase III

THREADED-C
Main Features of EARTH

- Fast thread context switching
- Efficient parallel function invocation
- Good support of fine-grain dynamic load-balancing
- Efficient support of split-phase transactions and fibers

Note—Items marked with a \( \Delta \) are features unique to EARTH, and not found in the original Cilk model
Outline

1 Introduction
   • Secret Origins of EARTH

2 The EARTH Program Execution Model

3 The EARTH Abstract Machine Model
   • The EARTH Abstract Machine

4 EARTH-Manna: An Implementation of the EARTH Architecture Model

5 Programming Models for Multithreaded Architectures
   • Features of Multithreaded Programming Models
   • EARTH Instruction Set
   • The EARTH Benchmark Suite (EBS)
   • Programming Examples
   • Compilation Environment, Revisited

6 Summary
Summary of EARTH-C

Extensions

- Explicit parallelism
  - Parallel vs. sequential statement sequences
  - forall loops
- Locality annotation
  - Local vs. remote memory references (global, local, replicate, ...)
- Dynamic load-balancing
  - Basic vs. remote function & invocation sites
References I


- Guang R. Gao and Vivek Sarkar (1995). “Location Consistency: Stepping Beyond the Memory Coherence Barrier”. In: ICPP (2), pp. 73–76
