Topic 4c:
A hybrid Dataflow-Von Neumann PXM:
The EARTH Experience

CPEG421/621: Compiler Design

Material mostly taken from Professor Guang R. Gao's previous courses, with additional material from J.Suetterlein.

Some Historical Perspective

- EARTH was a project initiated during the 90s.
- Many current architectural issues did not exist
- However the problem of efficiently hiding latencies (especially memory) has always been at the core of high-performance computing, where being too slow is considered a "functional bug"
- One path was explored using preemptive threads (e.g. POSIX threads)
- Another path used data flow models of computation applied to Von Neumann architectures, such as EARTH

Outline

- Part I: EARTH execution model
- Part II: EARTH architecture model and platforms
- Part III: EARTH programming models and compilation techniques
- The percolation model and its applications
- Summary

Part I

EARTH: An Efficient Architecture for Running THreads

[PACT95, EURO-PAR95, ICS95, MASCOTS96, ISCA96, PACT96, PPoPP97, PACT97, SPAA97, DIPES98, SPAA98 and many others …)

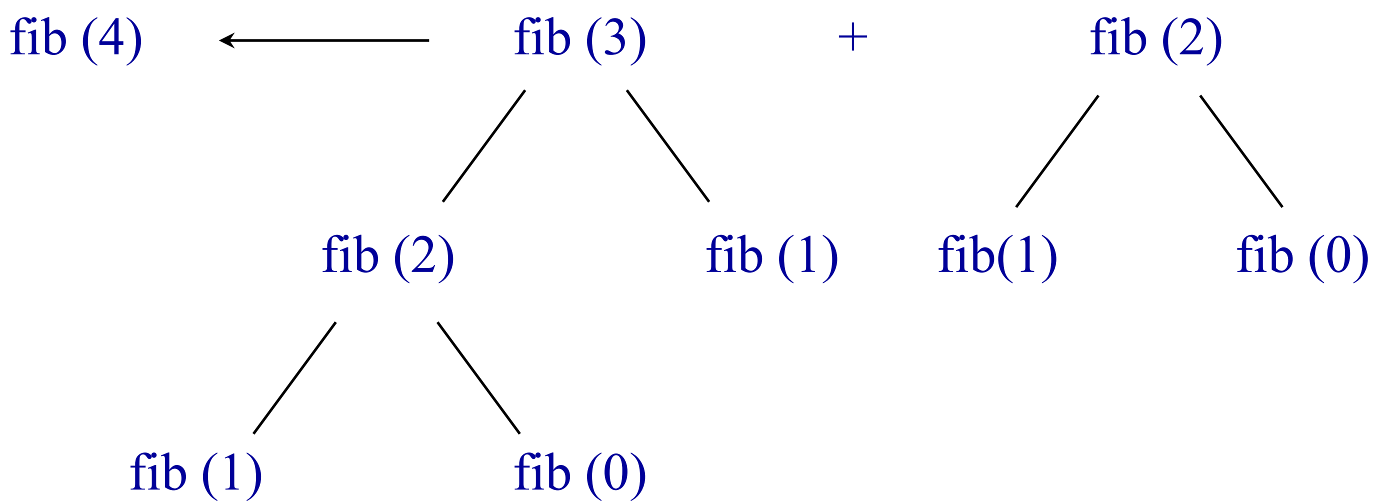
The EARTH Program Execution Model

- What is a thread?
- How the state of a thread is represented?
- How a thread is enabled?

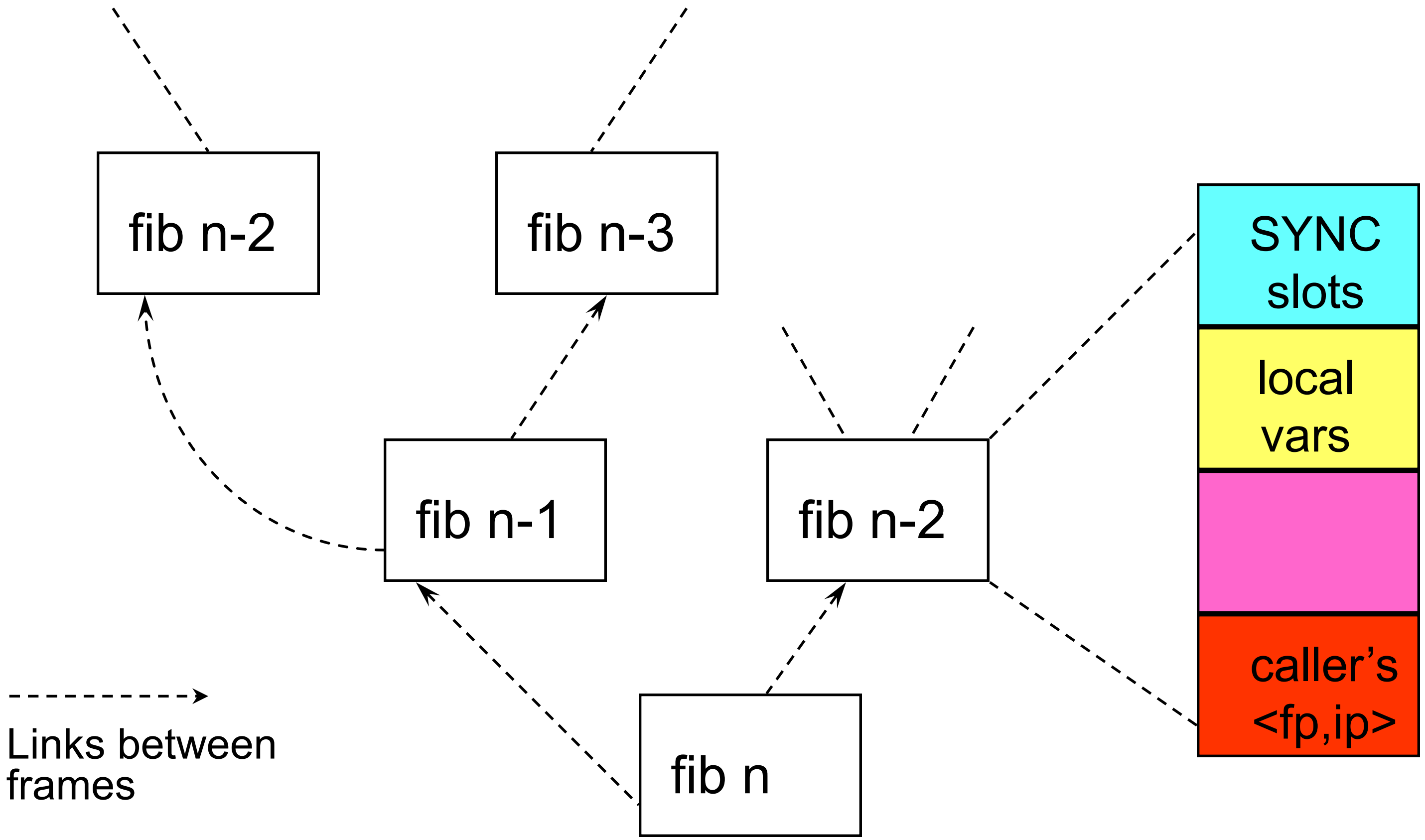
What is a Thread?

- A parallel function invocation *(threaded function invocation)*
- A code sequence defined (*by a user or a compiler*) to be a thread (*fiber*)
- Usually, a body of a threaded function may be partitioned into several threads

How to Execute Fibonacci Function in Parallel?



Parallel Function Invocation



Tree of "Activation Frames"

An Example

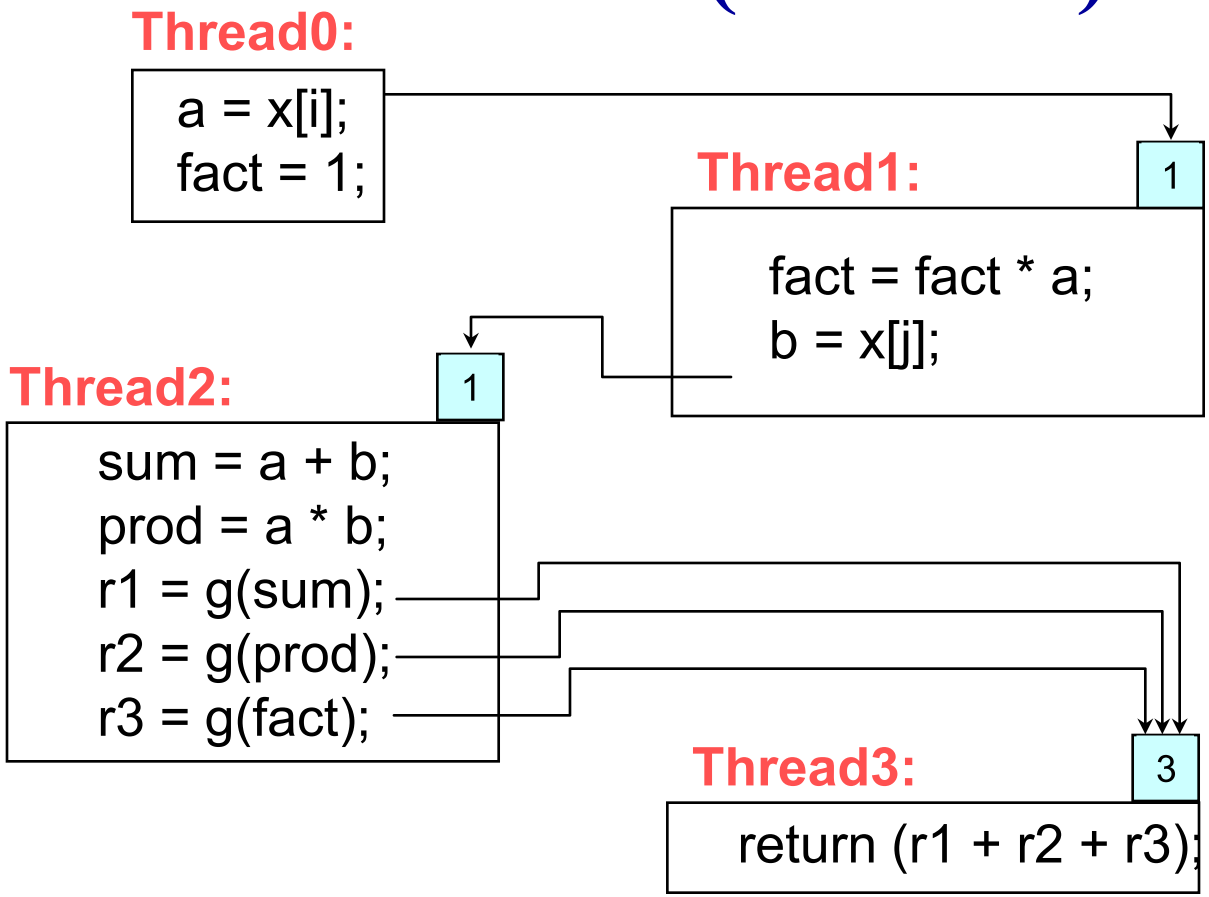
```
int f(int *x, int i, int j)
{
int a, b, sum, prod, fact;
int r1, r2, r3;
a = x[i];
fact = 1;
fact = fact * a;
```

```
b = x[j];
sum = a + b;
prod = a * b;

r1 = g(sum);
r2 = g(prod);
r3 = g(fact);

return(r1 + r2 + r3);
}
```

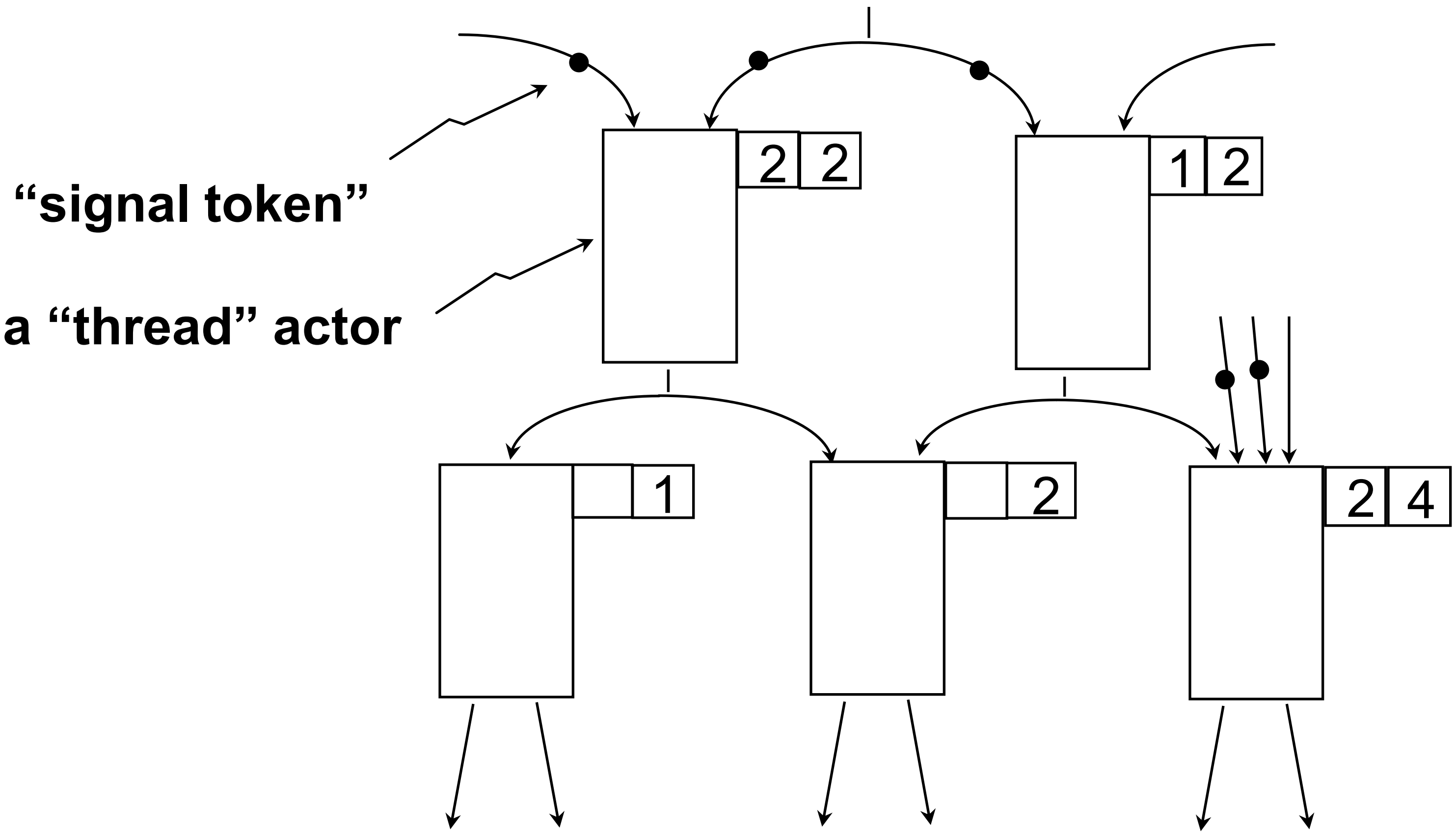
The Example is Partitioned into Four Fibers (Threads)



The State of a Fiber (Thread)

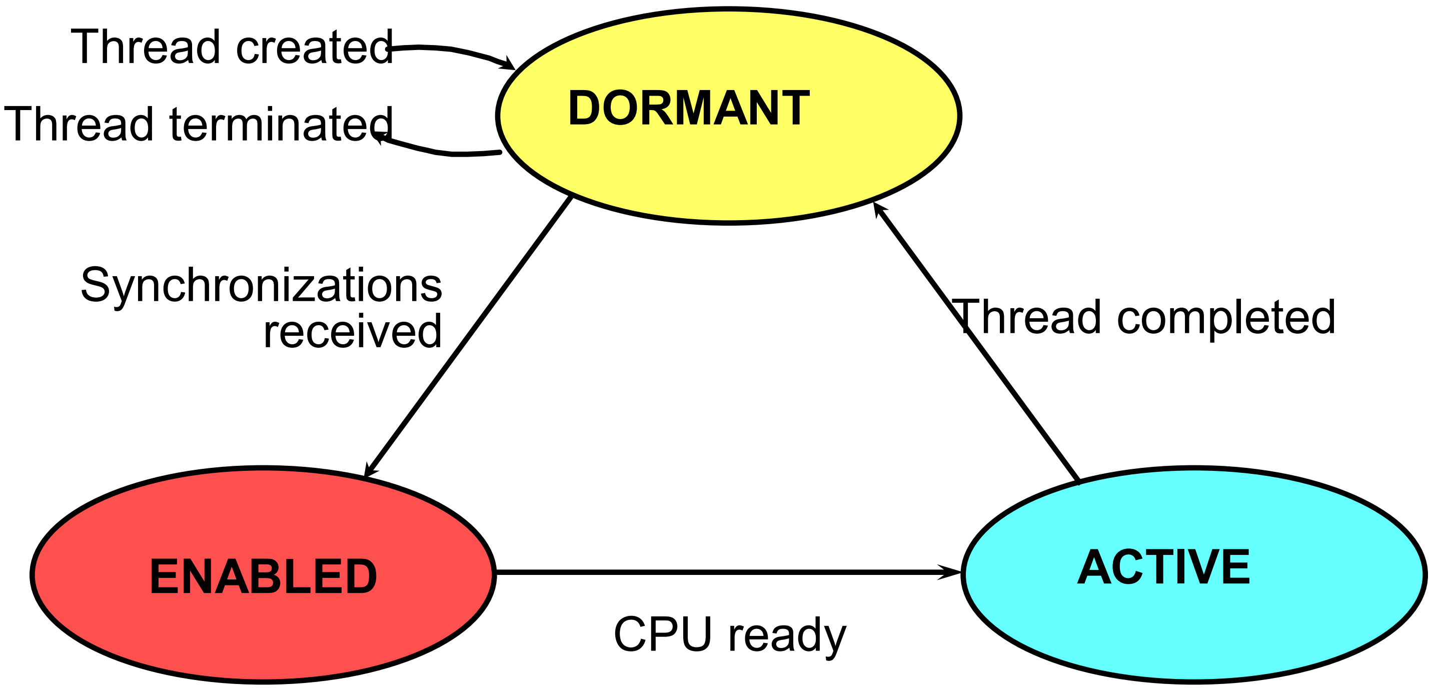
- A Fiber shares its "enclosing frame" with other fibers within the same threaded function invocation.
- The state of a fiber includes
  - its instruction pointer
  - its "temporary register set"
- A fiber is "ultra-light weighted": it does not need dynamic storage (frame) allocation.
- Our focus: non-preemptive threads – called ***fibers***

The "EARTH" Execution Model

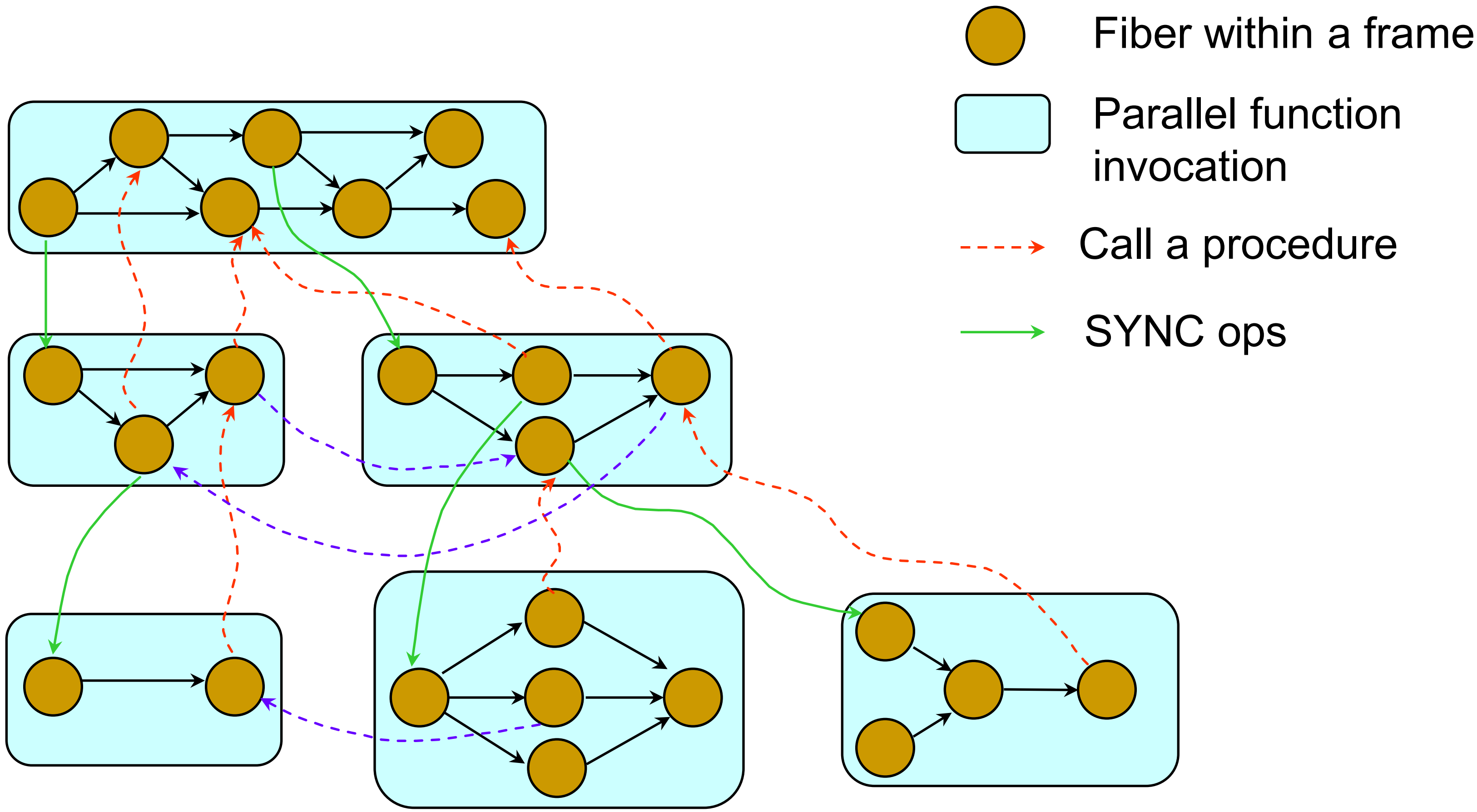


The EARTH Fiber Firing Rule

- 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.



Thread States



The EARTH Model of Computation

The EARTH Multithreaded Execution Model

Two Level of Fine-Grain Threads:

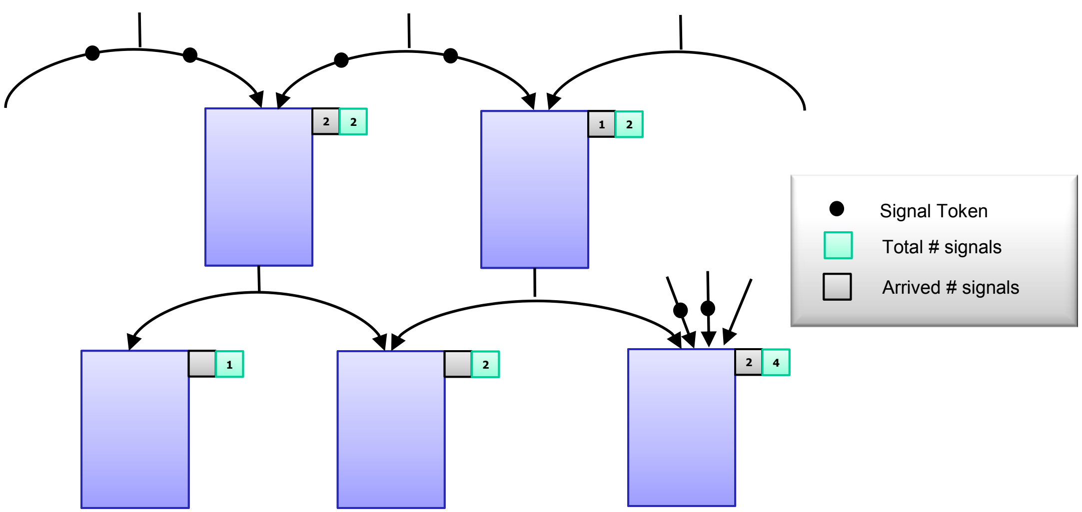
- threaded procedures
- fibers

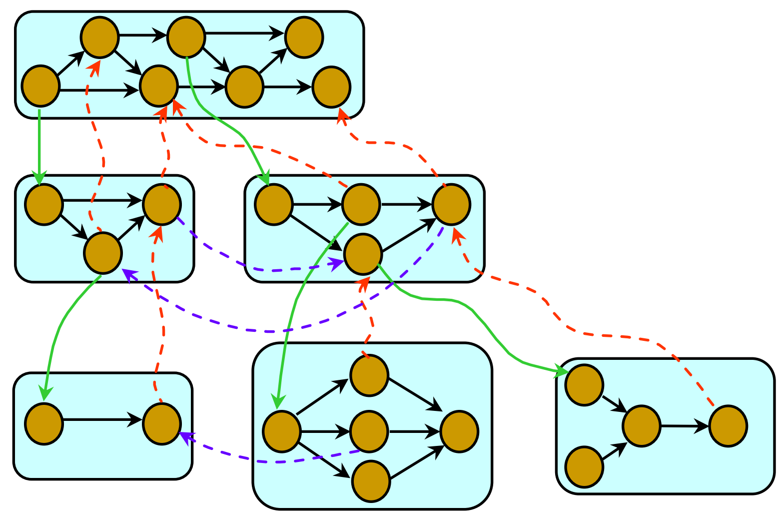
fiber within a frame

Aync. function invocation

A sync operation

Invoke a threaded func





A Side-Note: the Cilk Model
(thanks to J.Suetterlein)

- What is Cilk?
  - "A C language for programming dynamic multithreaded applications on shared-memory multiprocessors." –Leiserson, Lecture 1 http://supertech.csail.mit.edu/cilk/lecture-1.pdf
  - Is it a program execution model?
    - Three components
      - Threading, Memory, and Sync. Model
    - Throughout the literature all three components are discussed
      - BUT Leiserson considers it a language…

The Cilk Language
(Thanks to J.Suetterlein)

REGULAR C

```
int fib (int n)
{
   if (n<2)
        return (n);
   else
   {
        int x,y;
        x = fib(n-1);
        y = fib(n-2);
        return (x+y);
   }
}
```

Cilk C

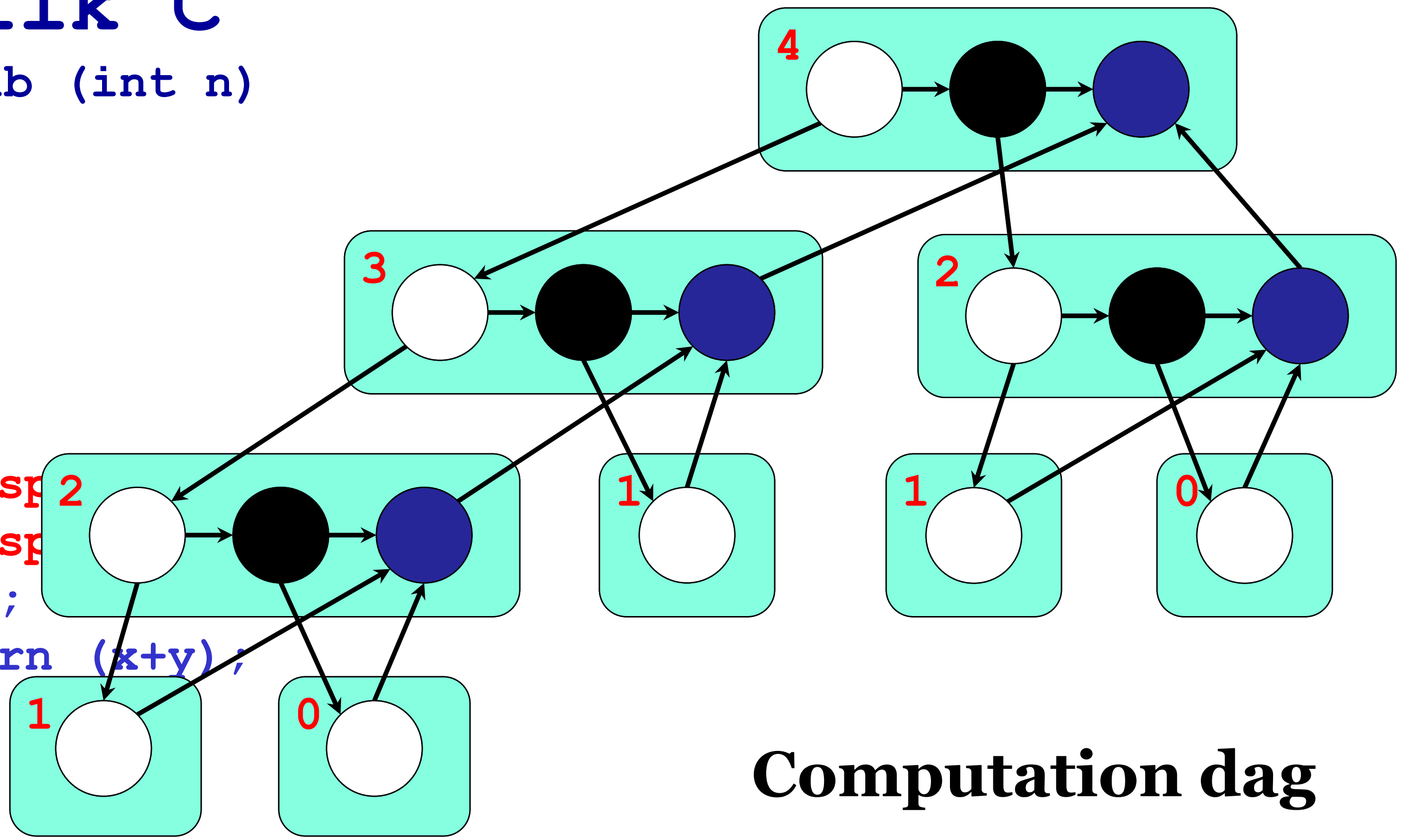
```
cilk int fib (int n)
{
   if (n<2)
        return (n);
   else
   {
        int x,y;
        x = spawn fib(n-1);
        y = spawn fib(n-2);
        sync;
        return (x+y);
   }
}
```

Fibonacci
(Thanks to J.Suetterlein)

Cilk C

```
cilk int fib (int n)
{
        sp
    y = sp
    sync;
    return (x+y);
  }
}
```

Example fib 4



Computation dag

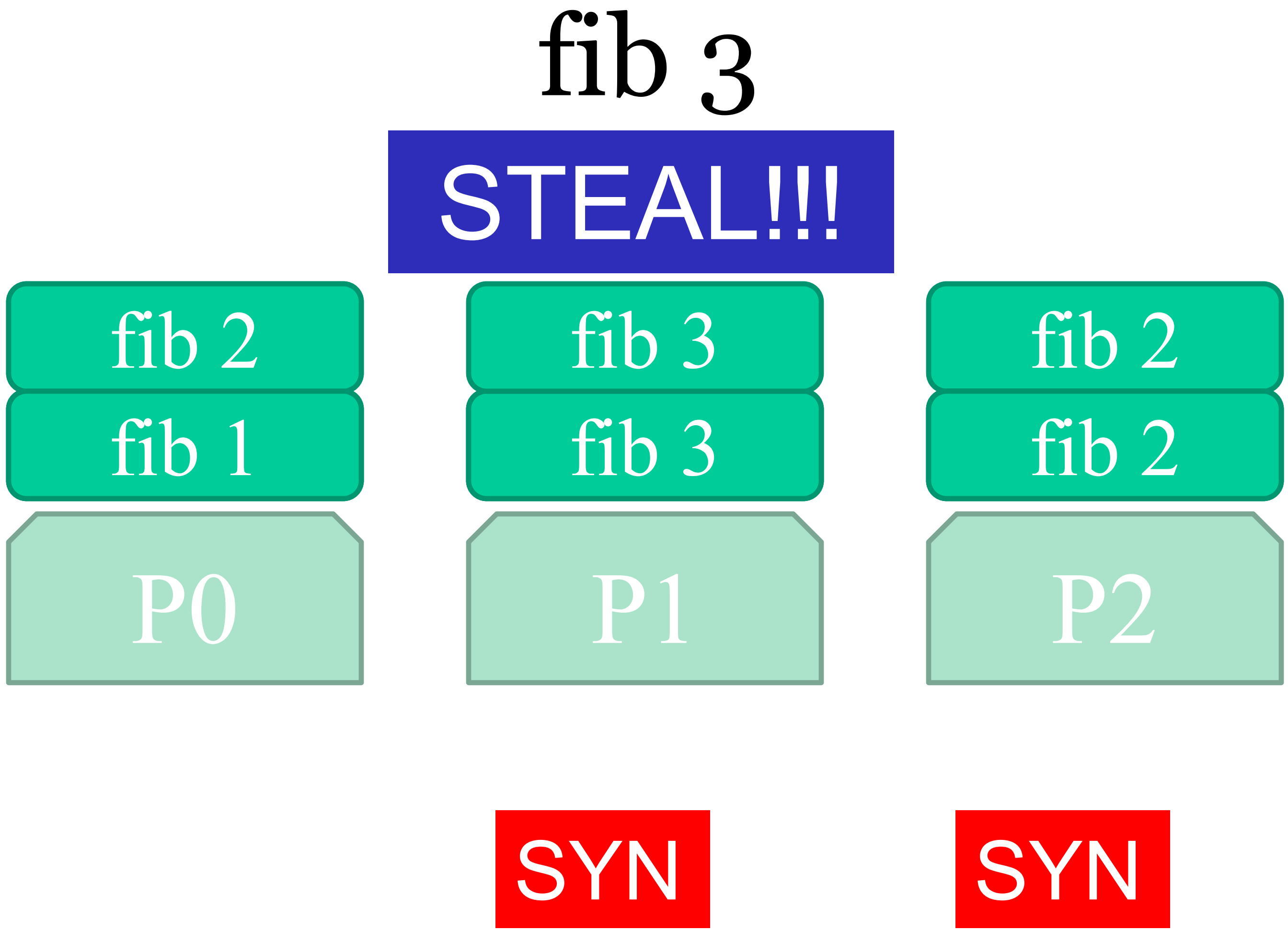
Scheduling – Cilk

(Thanks to J.Suetterlein)

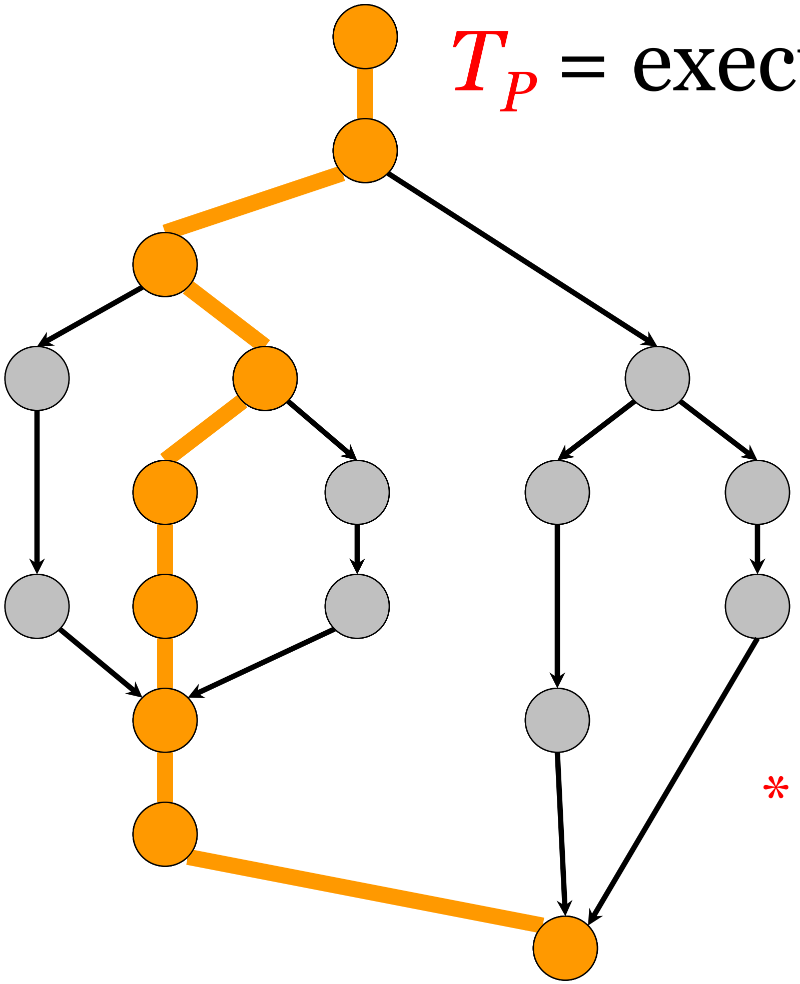
- Cilk's scheduler is greedy!
- **Work Stealing**
  - Each worker maintains a **deque** (double ended queue)
  - The worker pops and pushes work locally to the *bottom* of their own deque
  - When no work is available, the worker steals at *random* from the *top* of another workers deque
- On a spawn
  - Worker pushes parent to the bottom of the deque and begins working on the child
- On a sync
  - A sync maps to a "continuation closure" which contains a counter. The continuation will not be scheduled until all its dependencies are met (pg 60 section 5.1 of Blumof's PhD thesis).

Scheduling – Example

(Thanks to J.Suetterlein)



Computation Dag
(Thanks to J.Suetterlein)



T_P = execution time on P processors

T_1 = work

T_∞ = span*

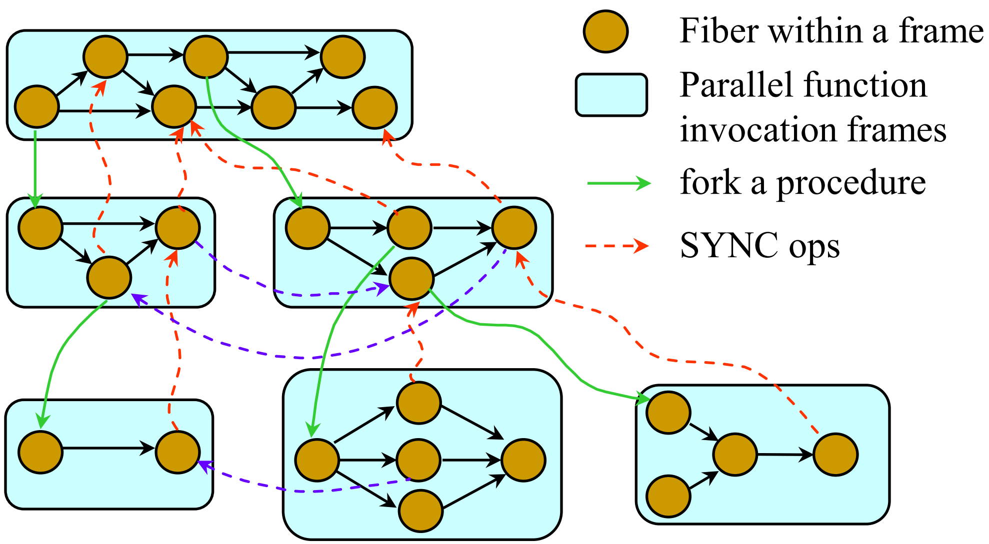
* Also called ***critical-path length*** or ***computational depth***.

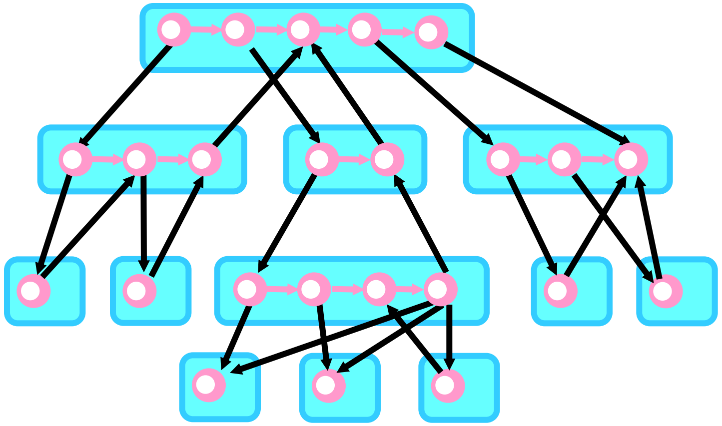
Cilk's Properties
(Thanks to J.Suetterlein)

Cilk offers several proofs w.r.t. space and time:

- Cilk's work-stealing scheduler achieves an expected running time of $T_p < \frac{T_1}{P} + O(T_\infty)$
- If a serial execution of Cilk program takes a stack space S_1, then the space required by a P-processor execution is at most $S_p \times PS_1$
  - Compare that with normal POSIX threads execution: is there any space and/or time guarantees?

EARTH vs. CILK



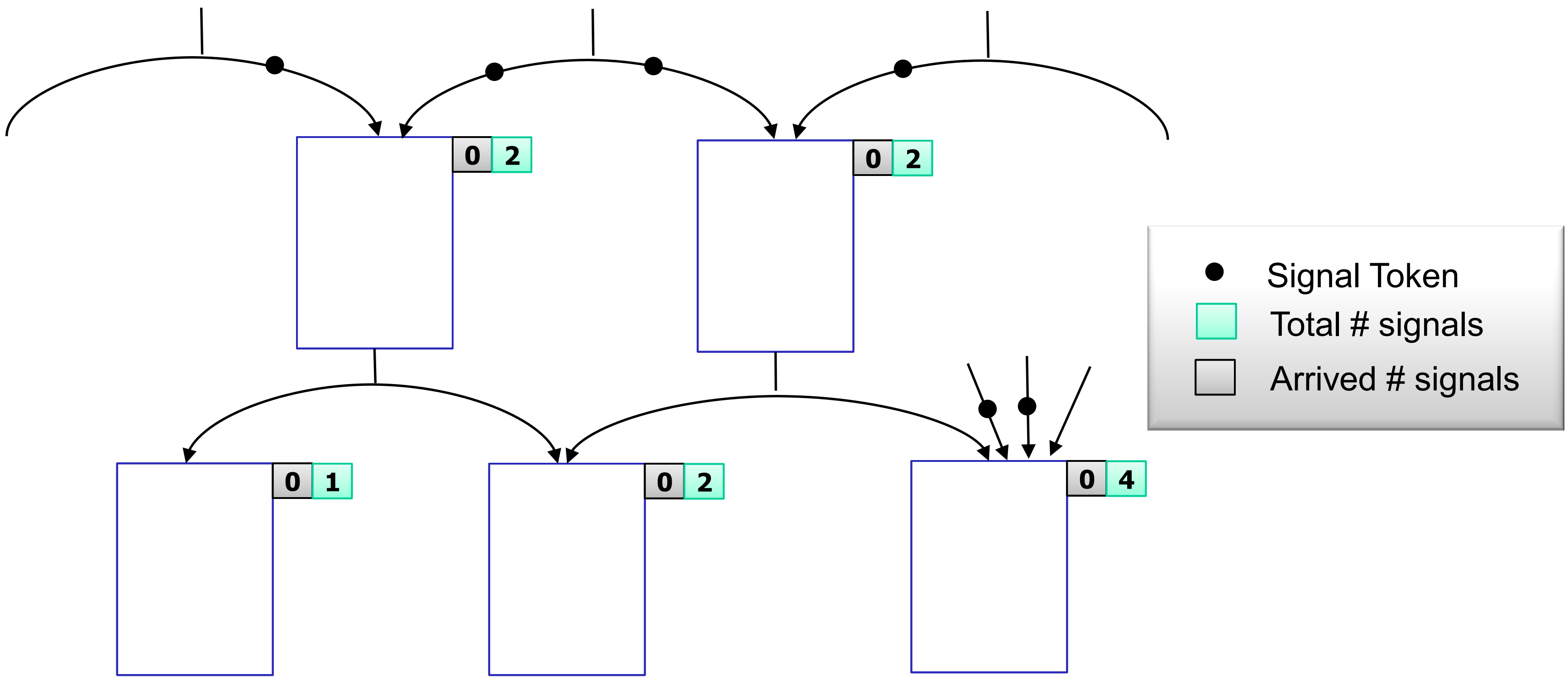


EARTH Model

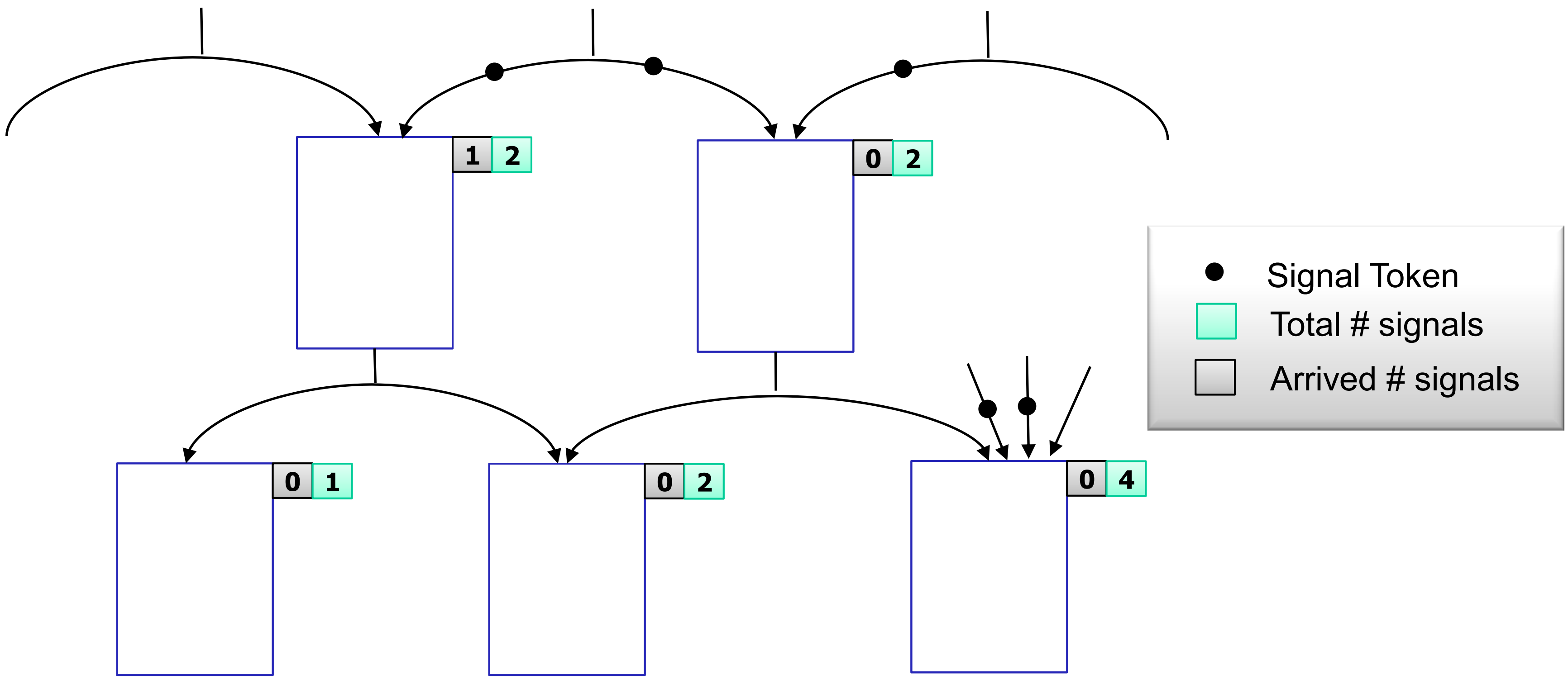
CILK Model

Note: EARTH has it origin in static dataflow model

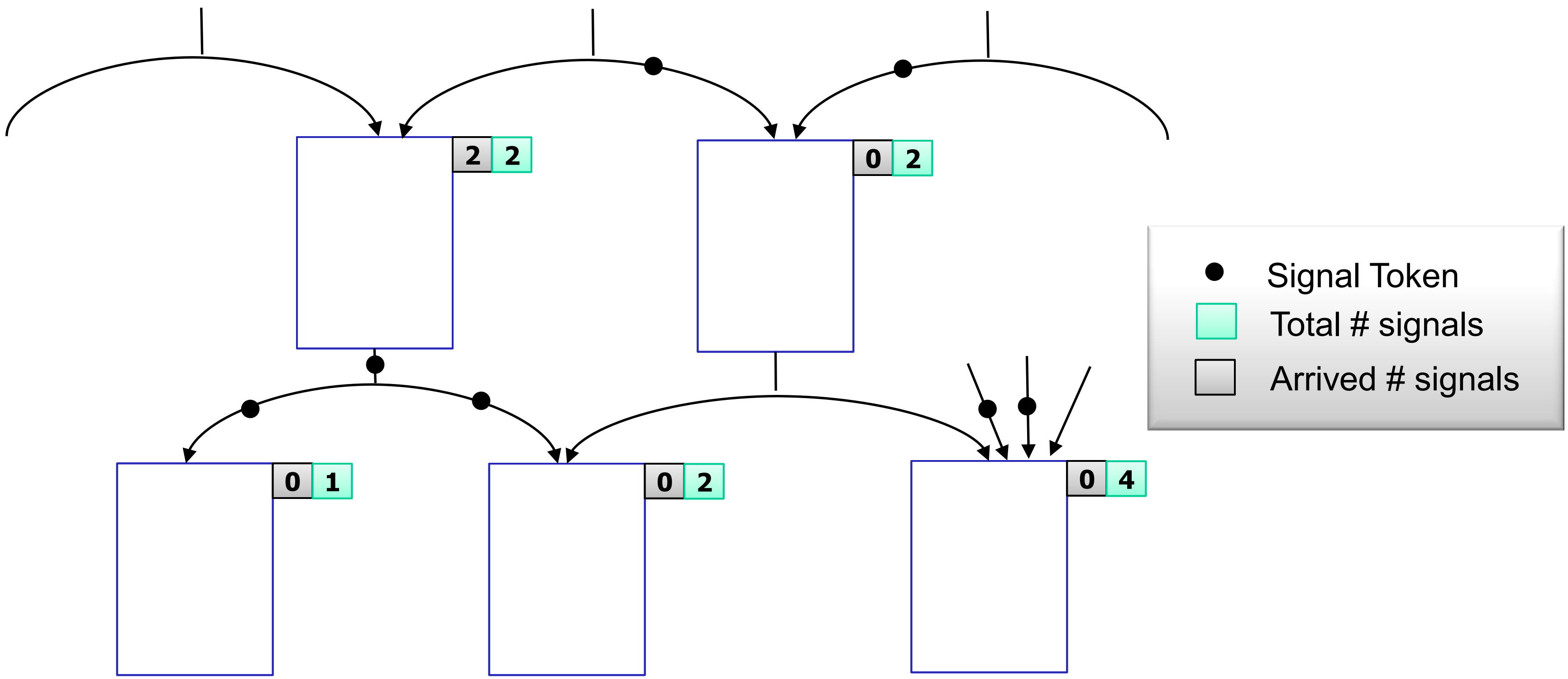
The "Fiber" Execution Model



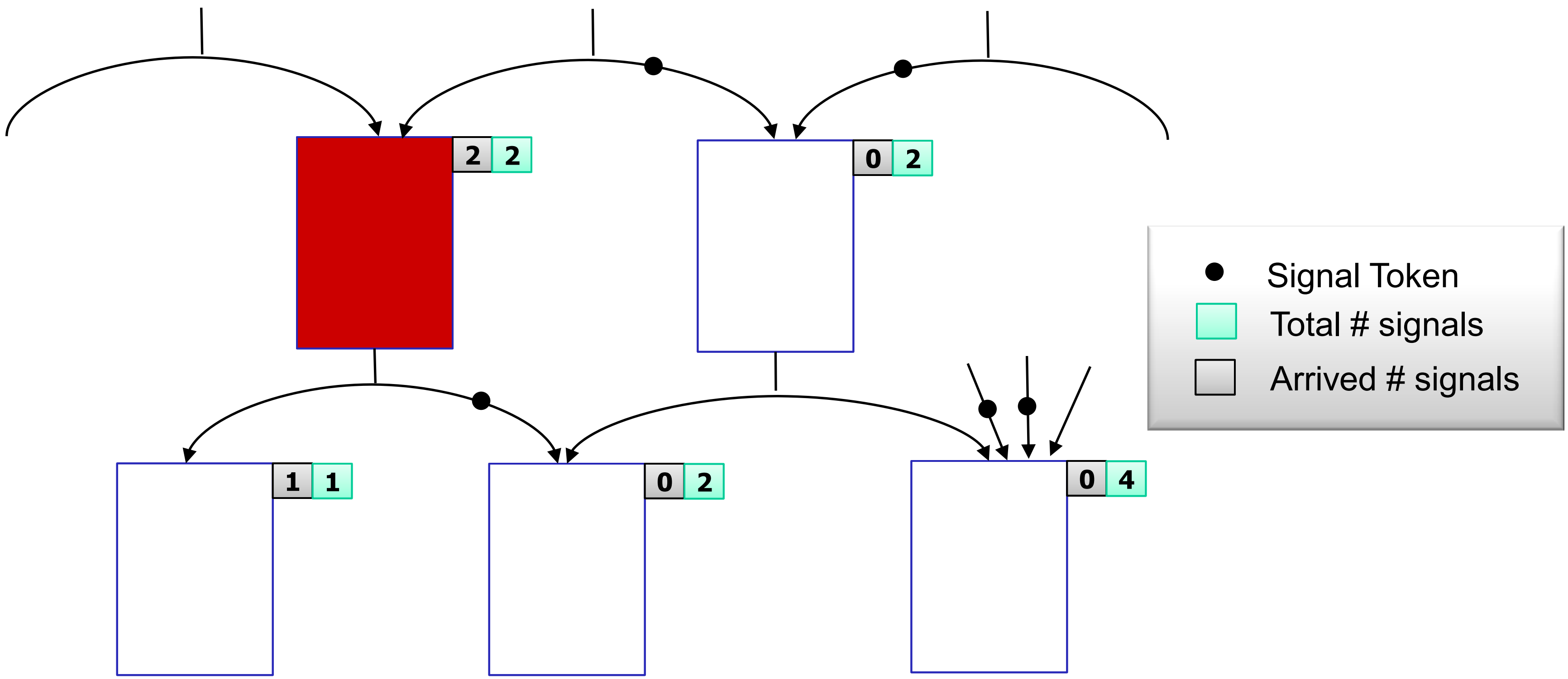
The “Fiber” Execution Model



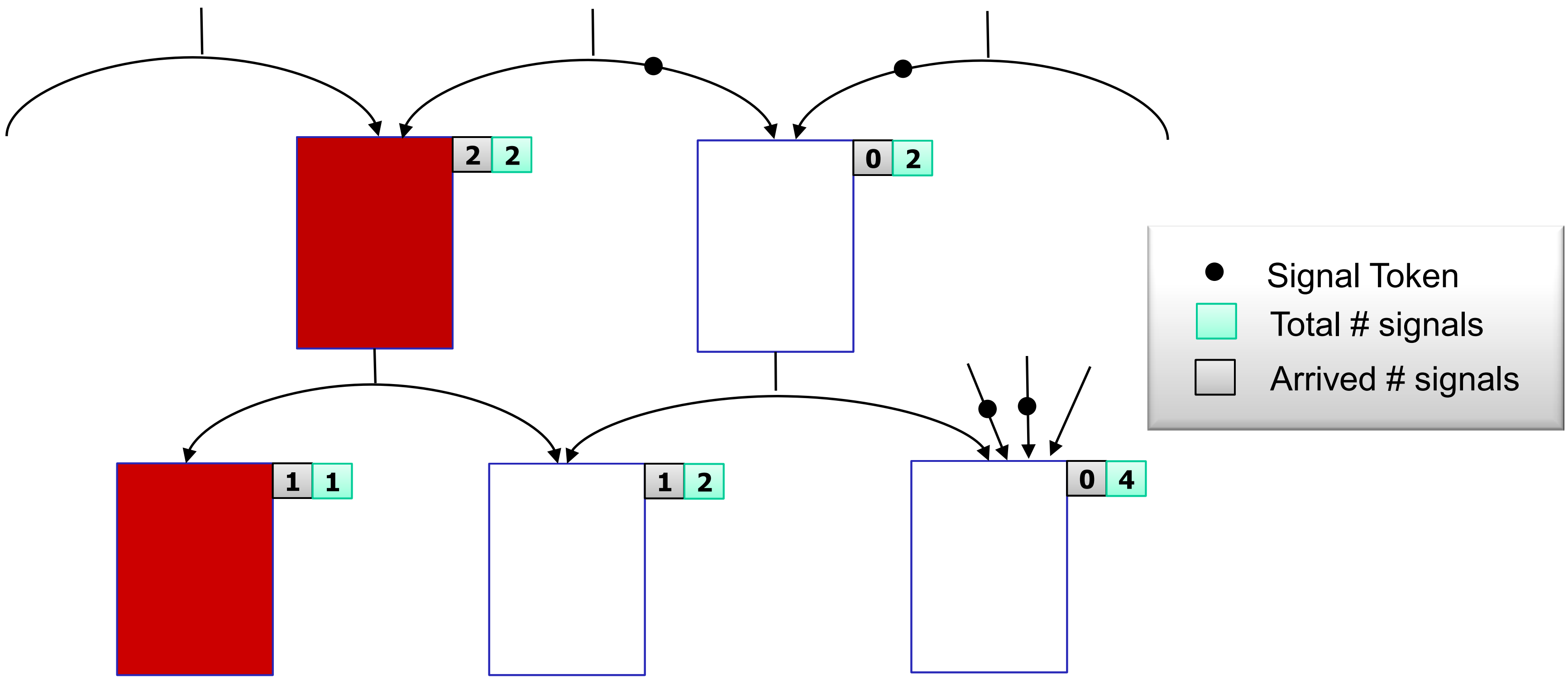
The "Fiber" Execution Model



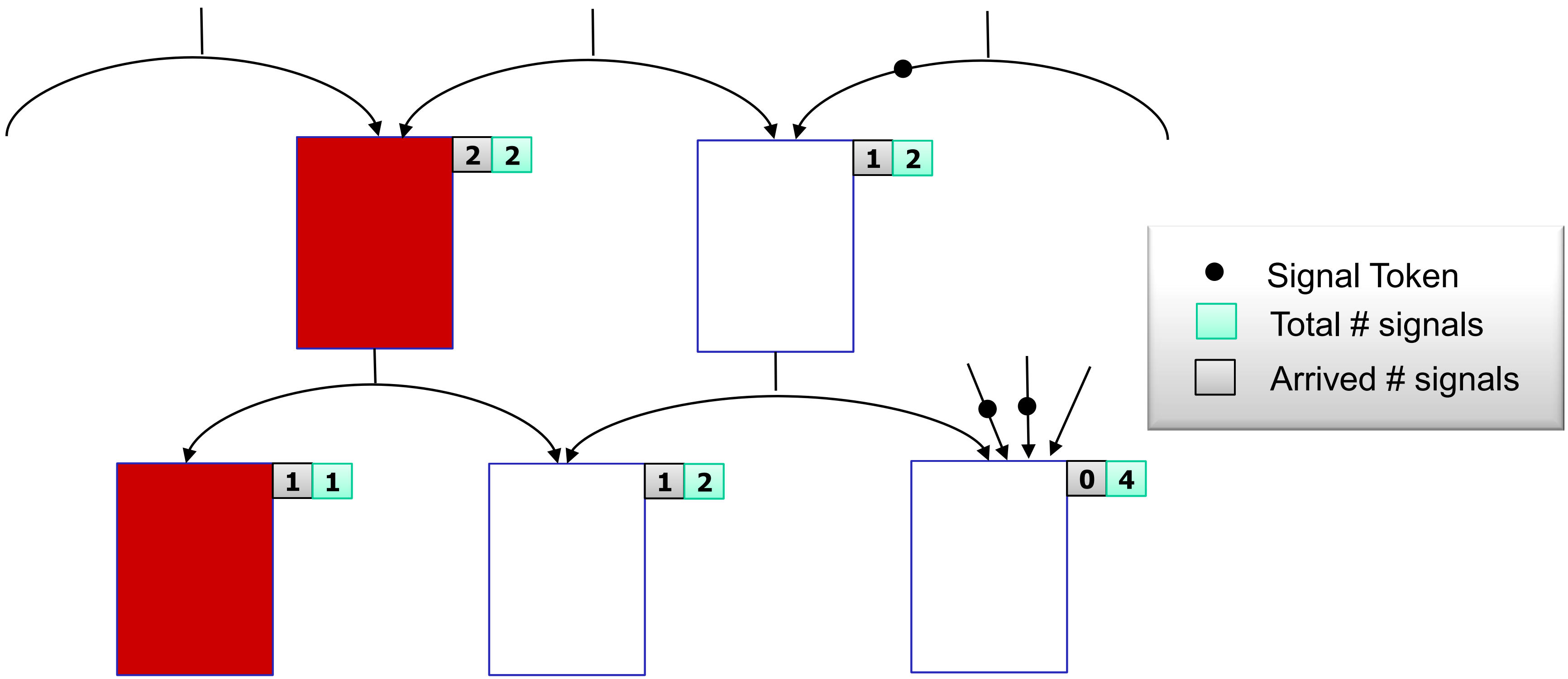
The "Fiber" Execution Model



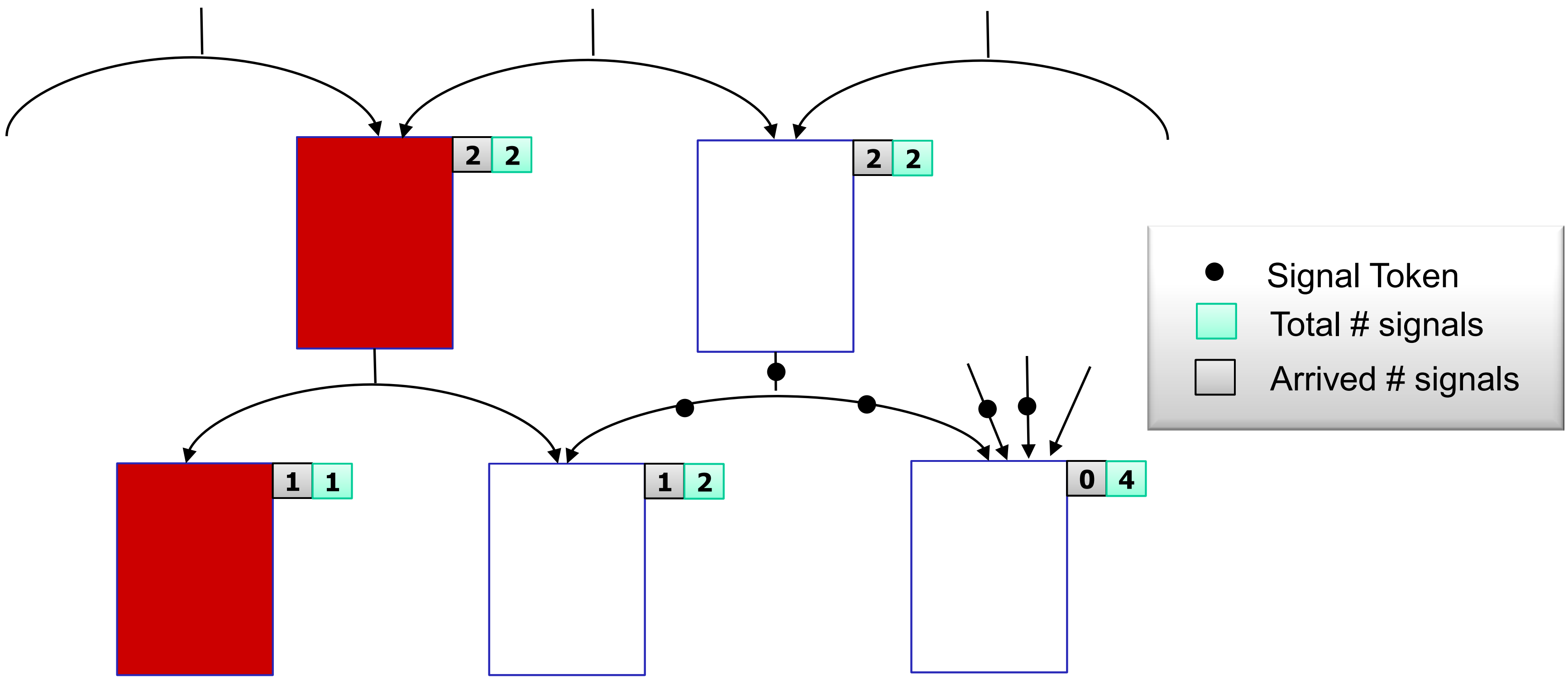
The “Fiber” Execution Model



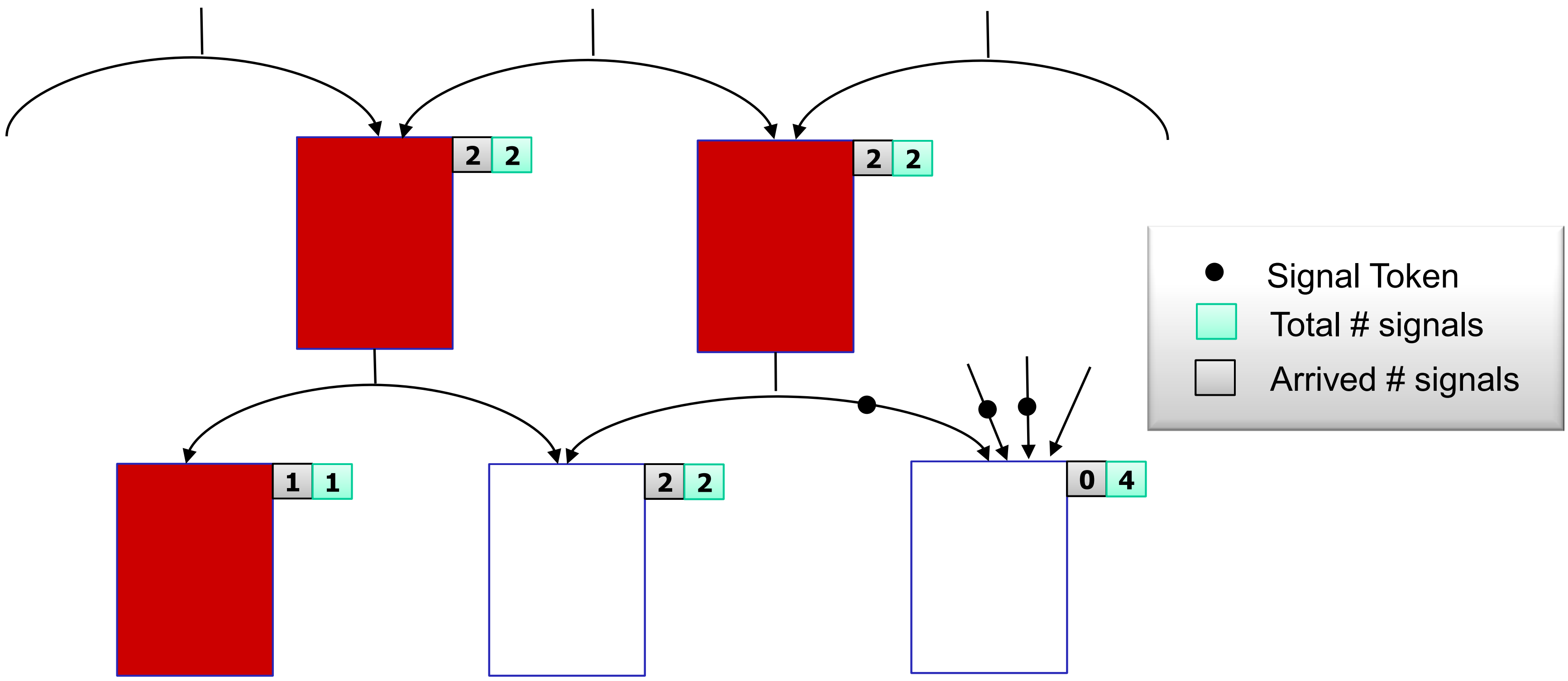
The “Fiber” Execution Model



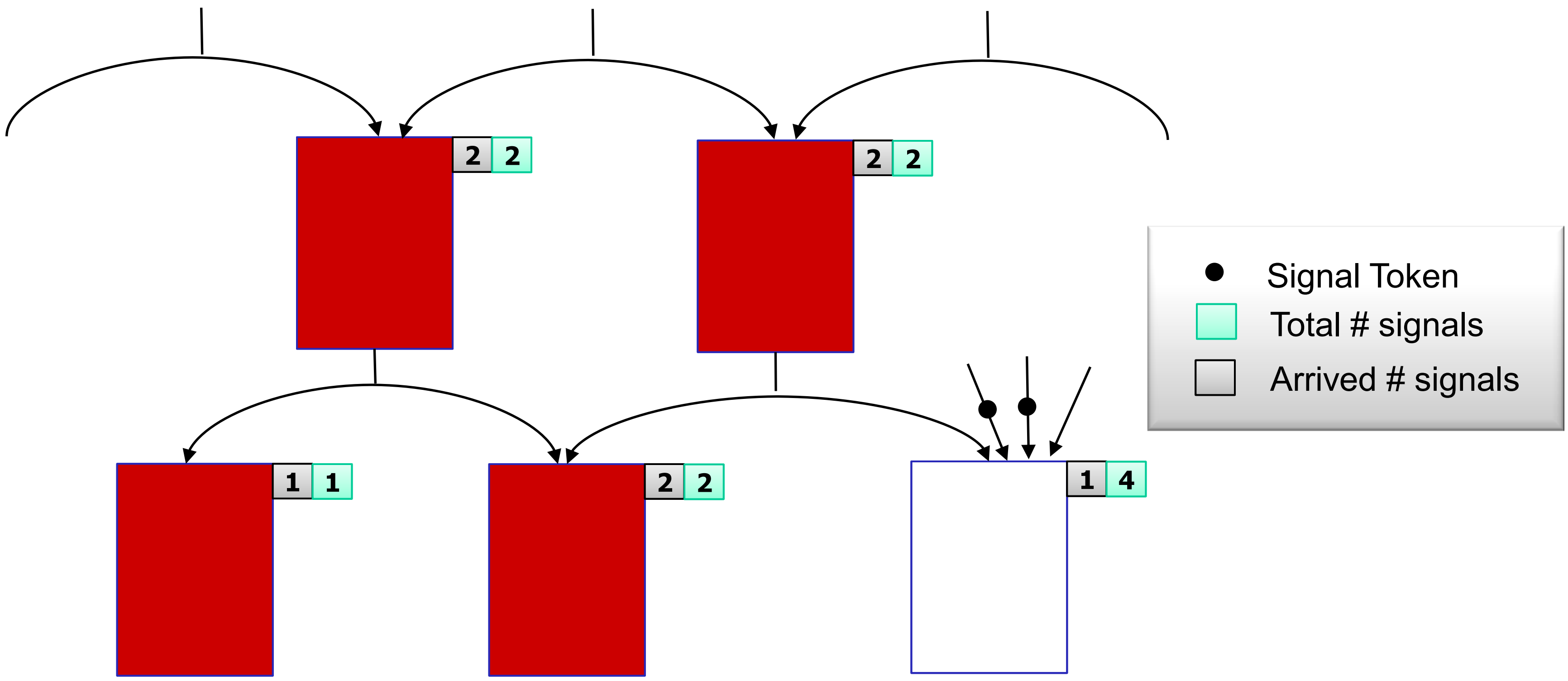
The “Fiber” Execution Model



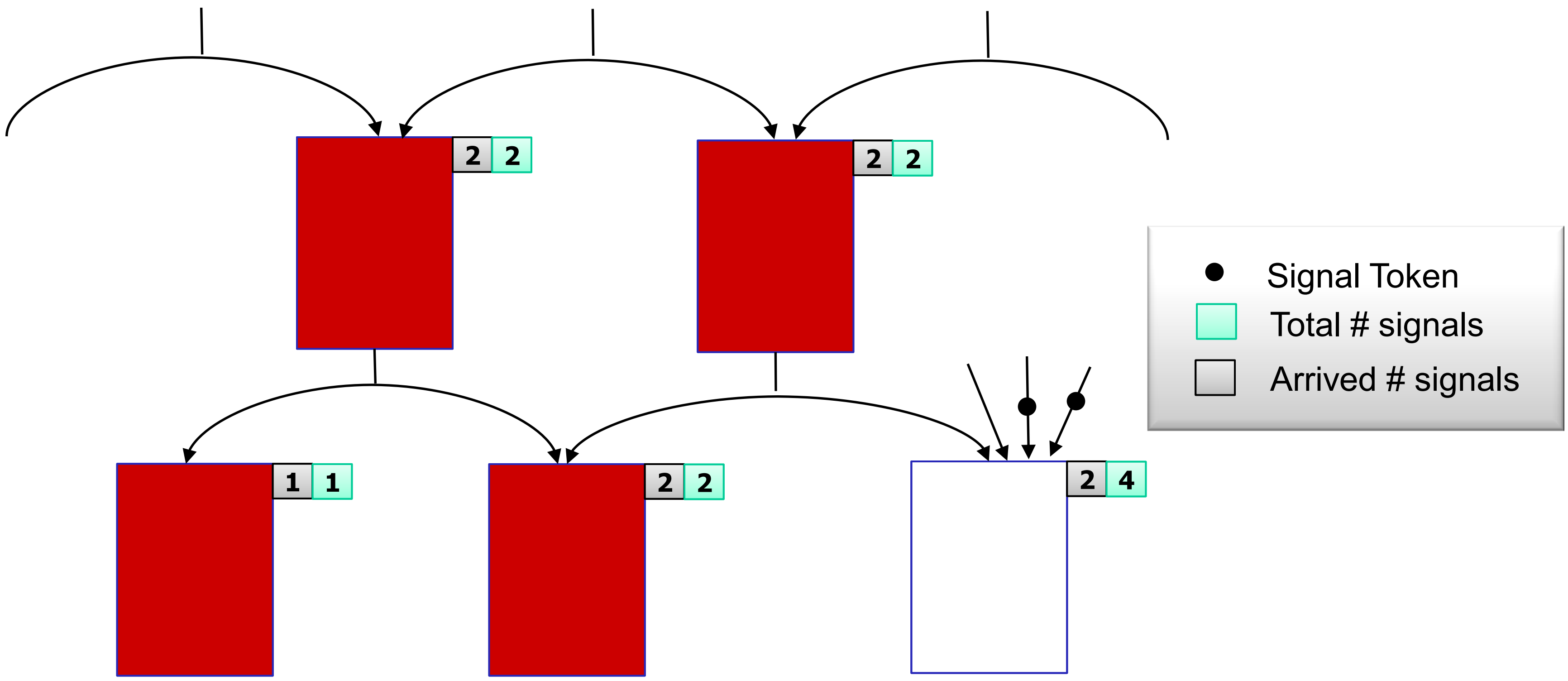
The “Fiber” Execution Model



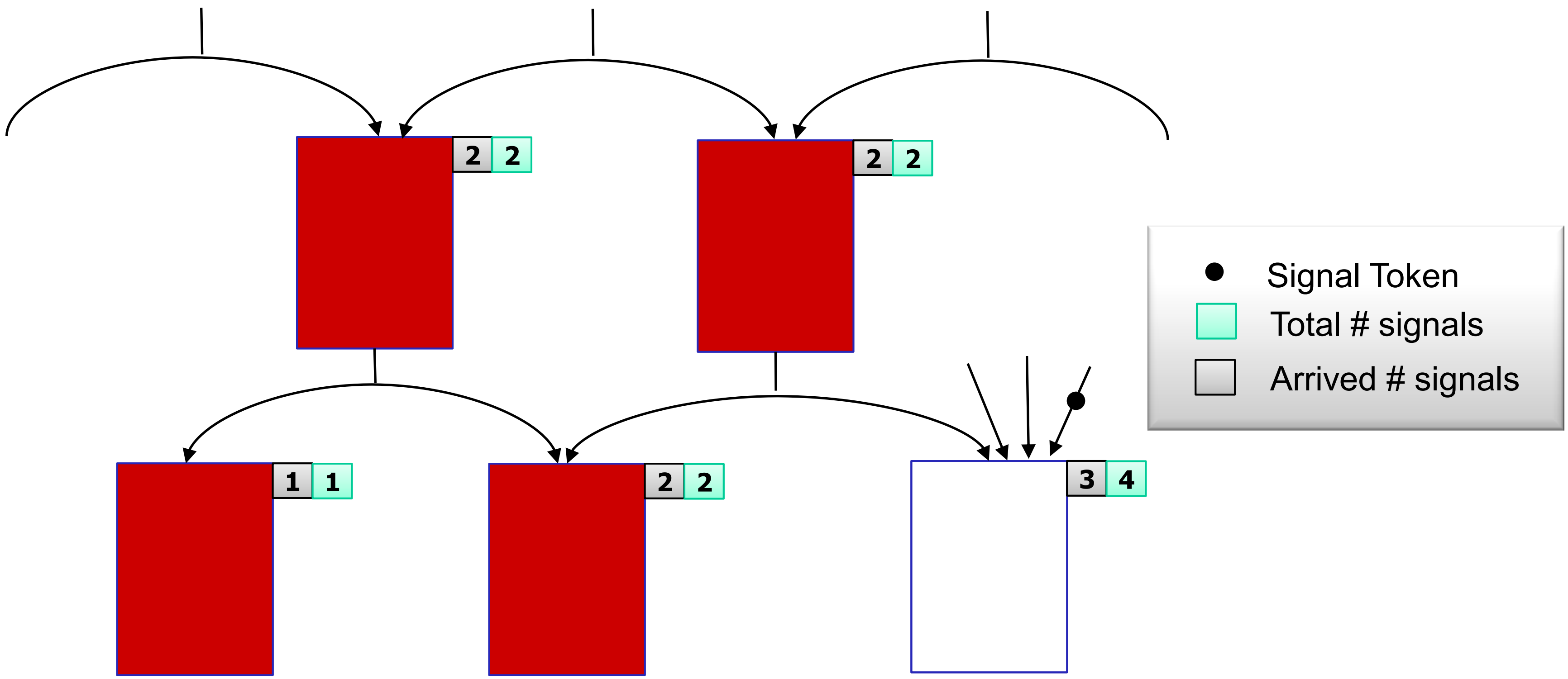
The “Fiber” Execution Model



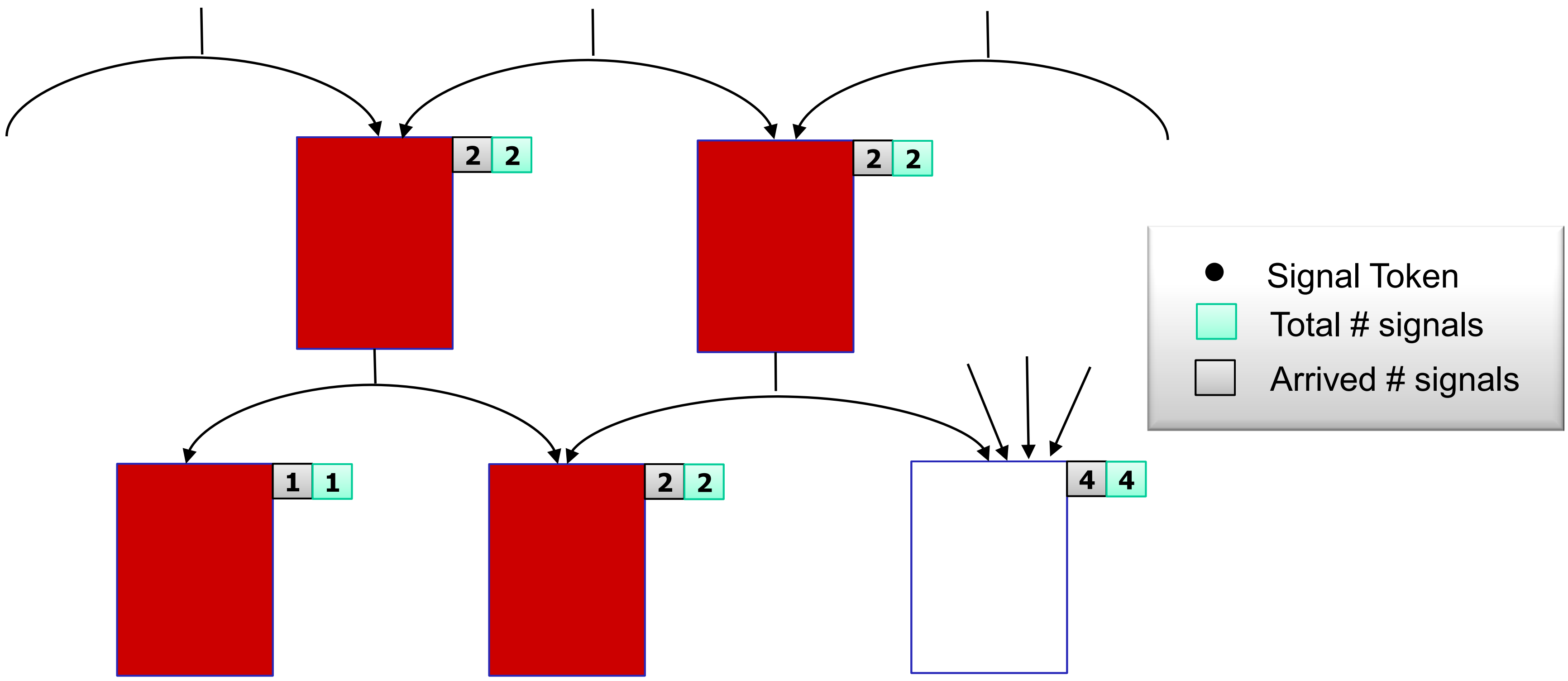
The “Fiber” Execution Model



The "Fiber" Execution Model

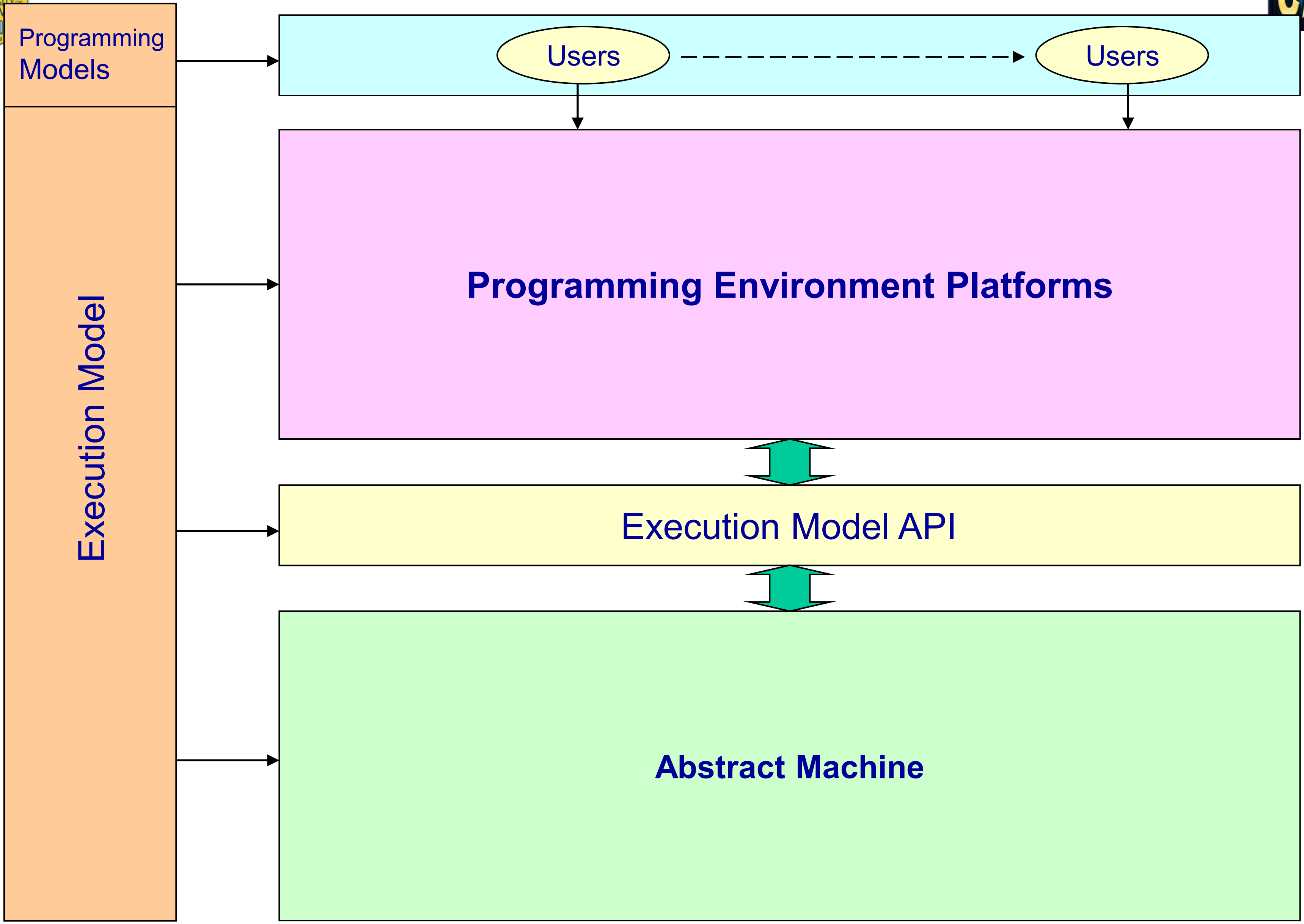


The "Fiber" Execution Model

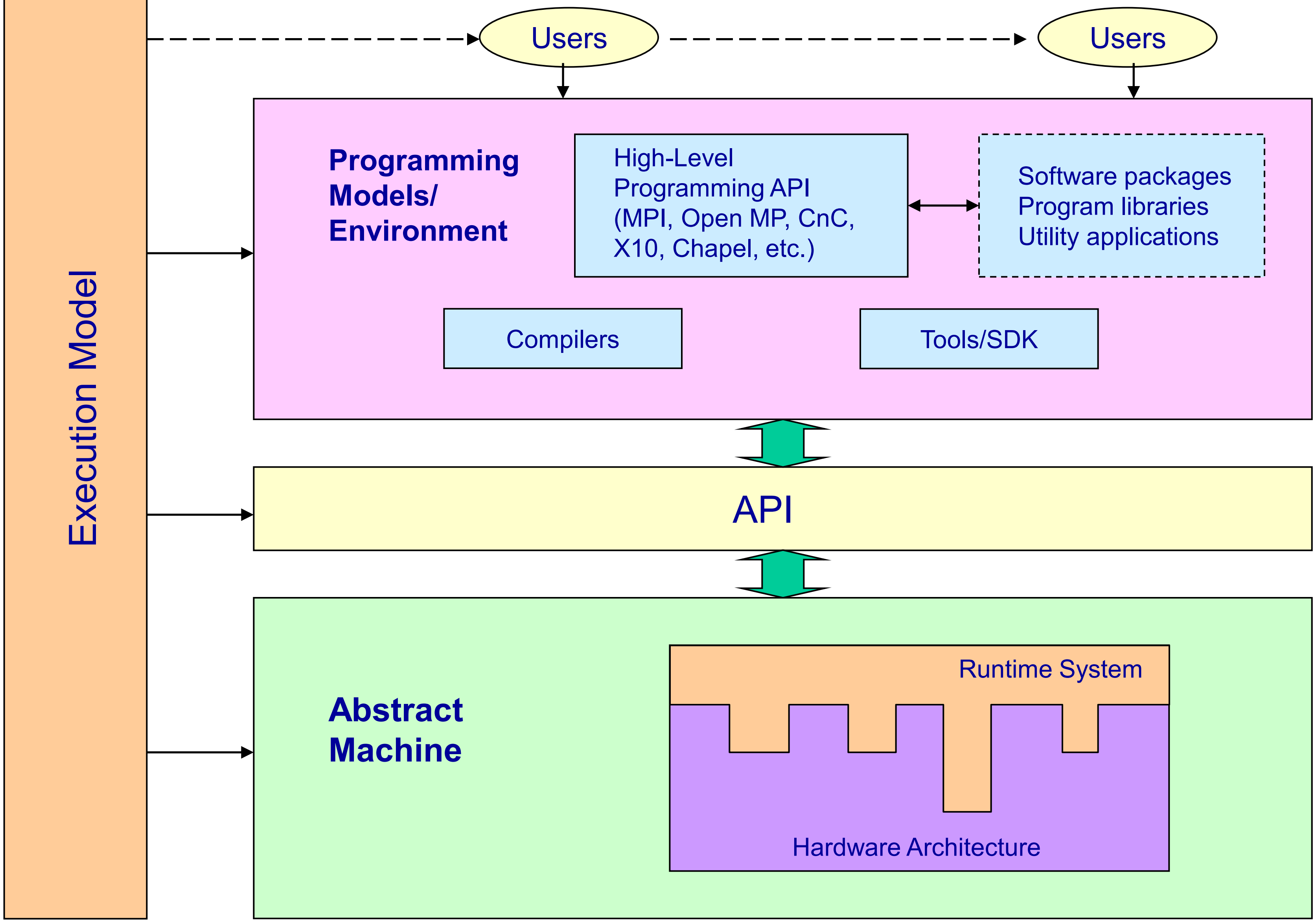


Part II

The EARTH
Abstract Machine (Architecture) Model
and
EARTH Evaluation Platforms

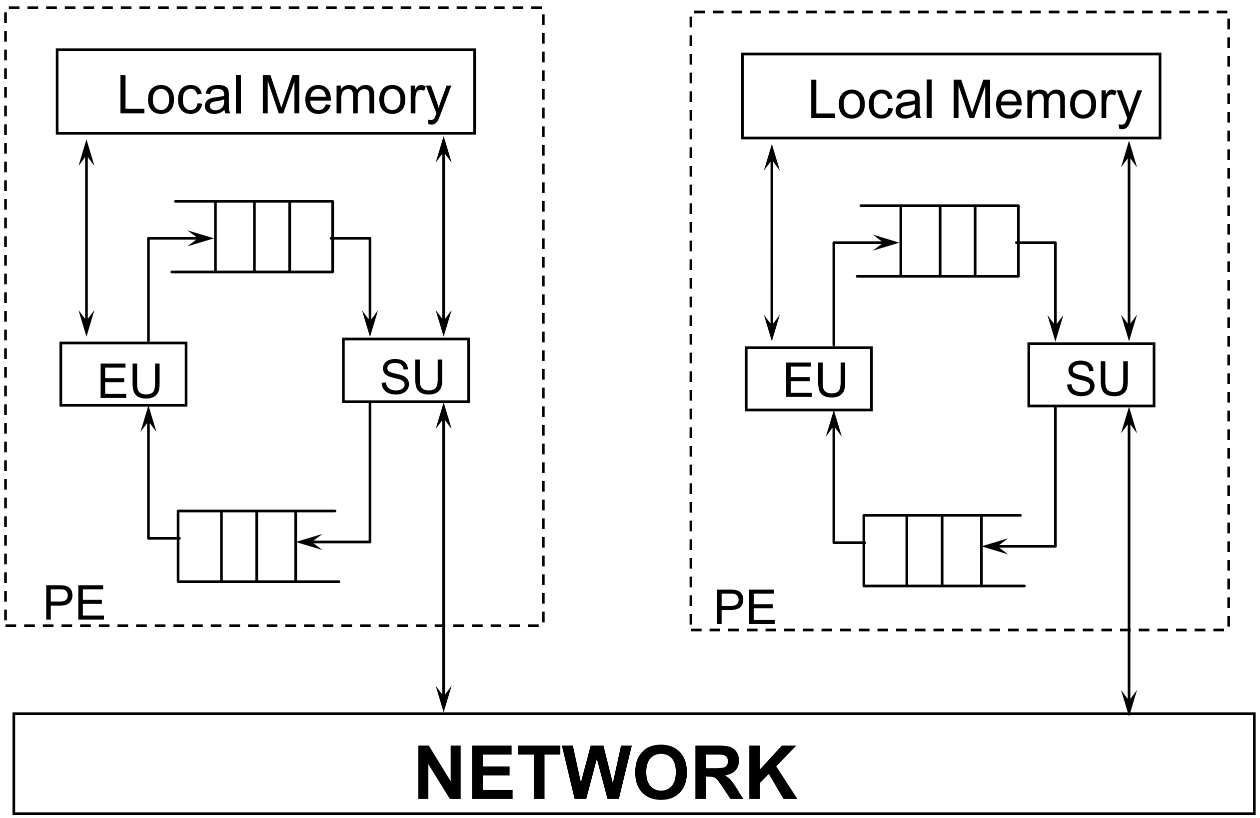


Execution Model and Abstract Machines



Execution Model and Abstract Machines

The EARTH Abstract Architecture
(Model)



How To Evaluate EARTH Execution and Abstract Machine Model ?

EARTH Evaluation Platforms

EARTH-MANNA

Implement EARTH on a ***bare-metal*** tightly-coupled multiprocessor.

EARTH-IBM-SP

Plan to implement EARTH on a off-the-shelf Comercial Parallel Machine (IBM SP2/SP3)

EARTH on Clusters

- EARTH on Beowulf
- Implement EARTH on a cluster of UltraSPARC SMP workstations connected by fast Ethernet

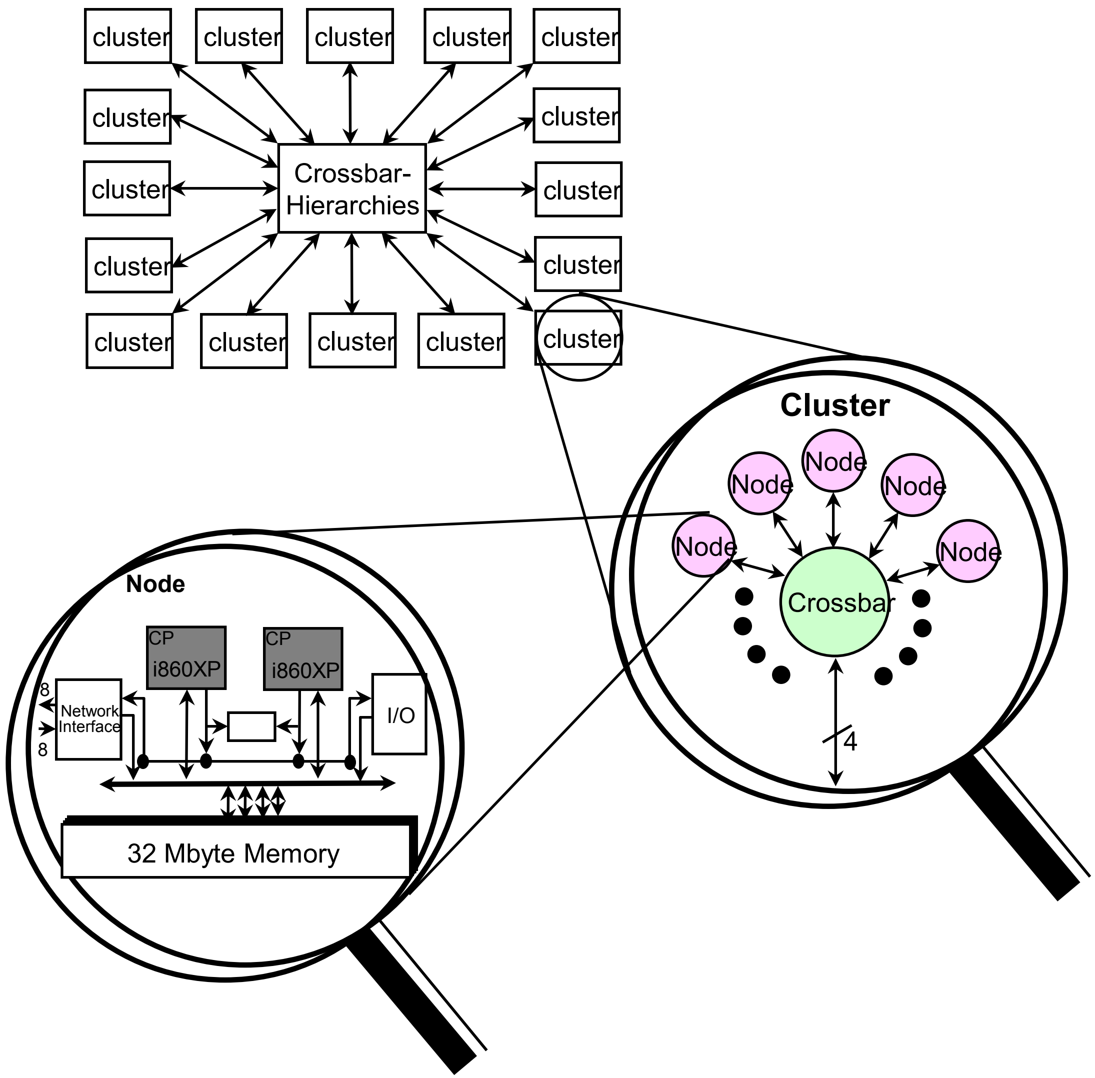
NOTE: ***Benchmark code are all written with EARTH Threaded-C: The API for EARTH Execution and Abstract Machine Models***

EARTH-MANNA: An Implementation of The EARTH Architecture Model

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 (both Class A & **B**)?

The EARTH-MANNA Multiprocessor Testbed

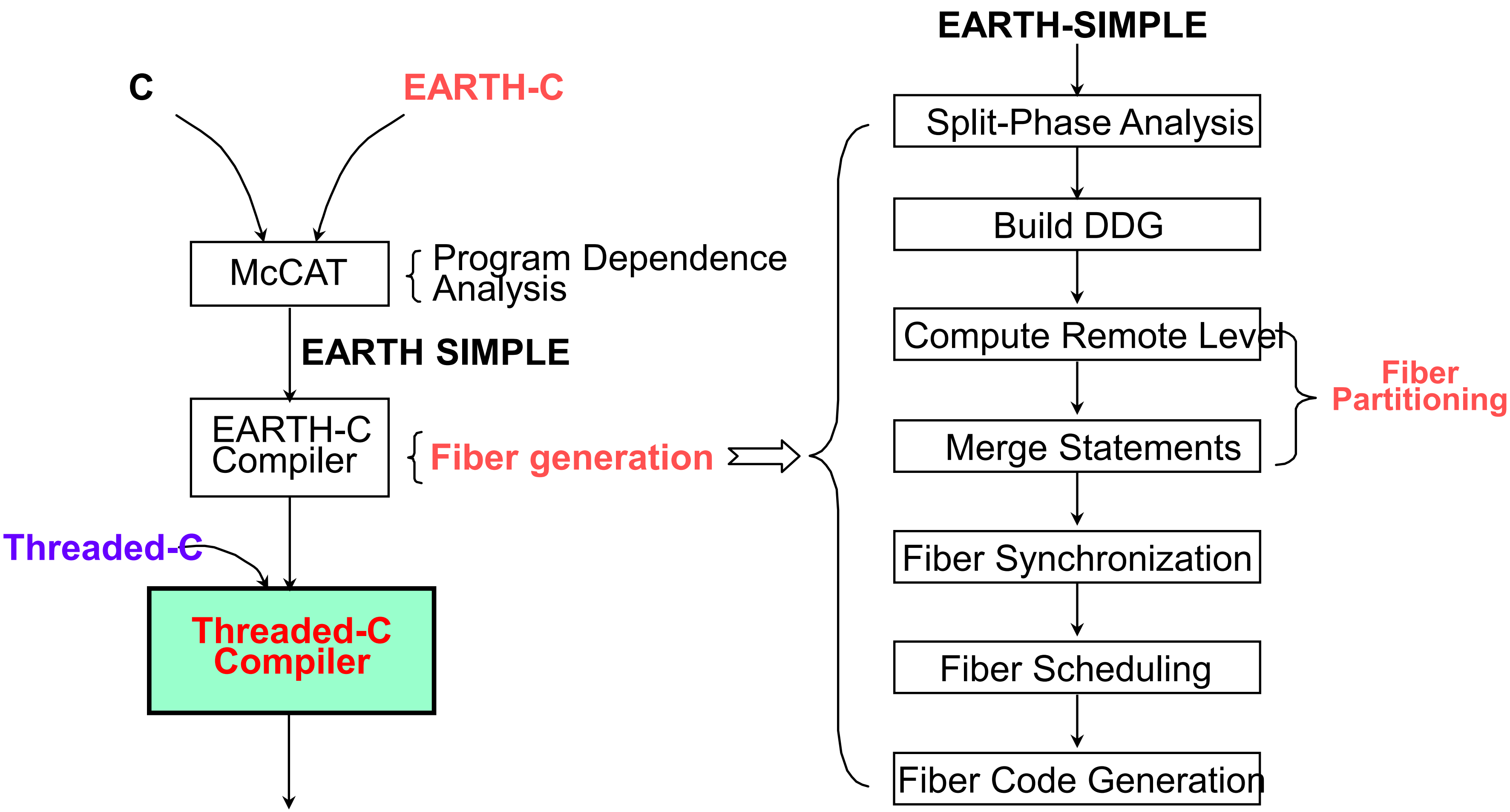


Main Features of EARTH Multiprocessor

Using off-the-shelf microprocessors

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

EARTH-C Compiler Environment



(a) EARTH Compilation Environment

(b) EARTH-C Compiler

Performance Study of EARTH

- Overview
- Performance of basic EARTH primitives
  (“Stress Test” via “micro-benchmarks”)
- Performance of benchmark programs
  - Speedup → $S = \frac{T_\infty}{T_1}$
  - USE value → $USE = T_{serial}/T_{EARTH-Serial}$
  - Latency Tolerance Capacity

NOTE: ***It is important to design your own performance “features” or “parameters” that best distinguishes your models from your counterparts***

Main Experimental Results of EARTH-MANNA

- Efficient multithreading support is possible with off-the-shelf processor nodes with overhead
  - context switch time ~ 35 instruction cycles
- A Multithread program execution model can make a big difference
  - Results from the EARTH benchmark suit (EBS)

Part III

Programming Models for Multithreaded Architectures:

The EARTH Threaded-C Experience

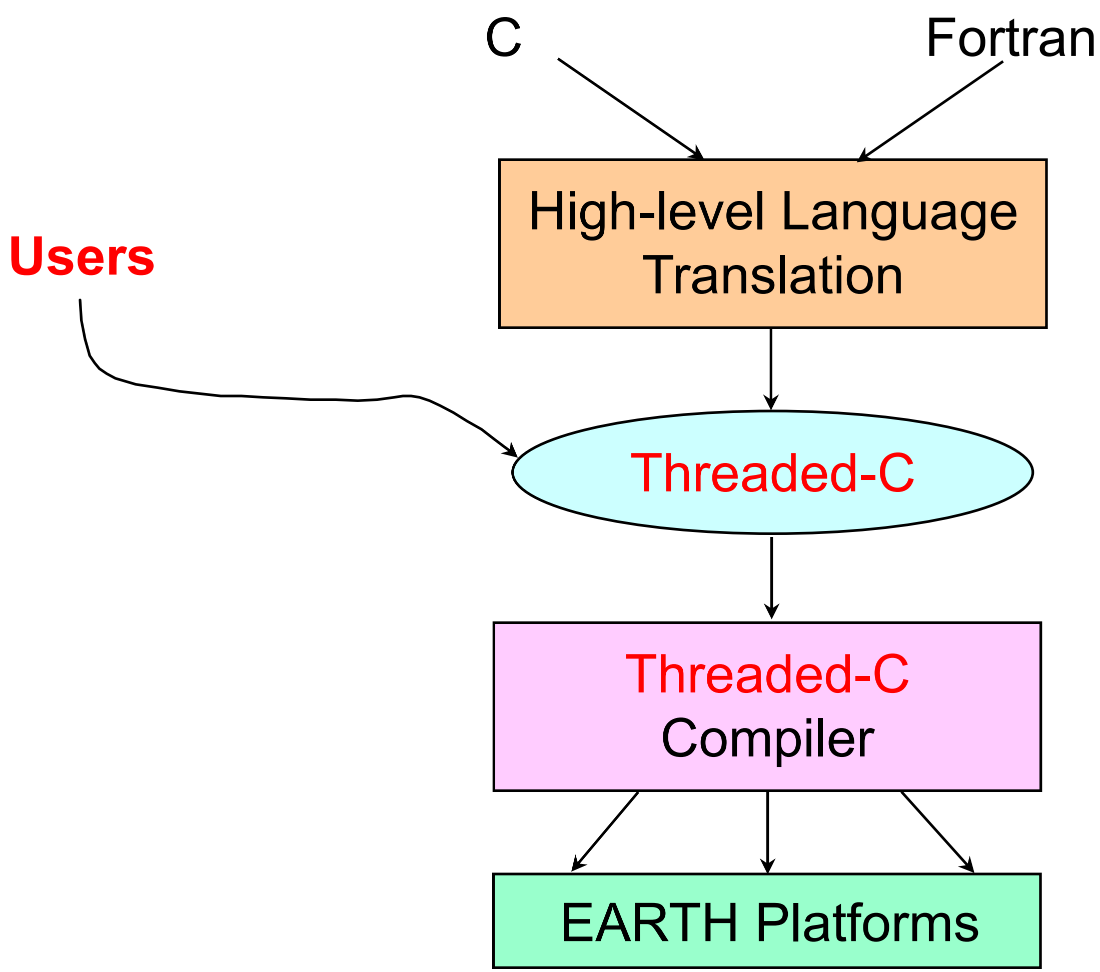
Outline

- Features of multithreaded programming models
- EARTH instruction set
- Programming examples

Threaded-C: A Base-Language

– To serve as a target language for high-level language compilers
– To serve as a machine language for EARTH architecture

The Role of Threaded-C



Features of Threaded Programming

Latency tolerance and management

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

The EARTH Operation Set

- The base operations
- Thread synchronization and scheduling ops

  SPAWN, SYNC
- Split-phase data & sync ops

  GET_SYNC, DATA_SYNC
- Threaded function invocation and load balancing ops

  INVOKE, TOKEN

Table 1. EARTH Instruction Set

- **Basic instructions**
  Arithmetic, Logic and 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

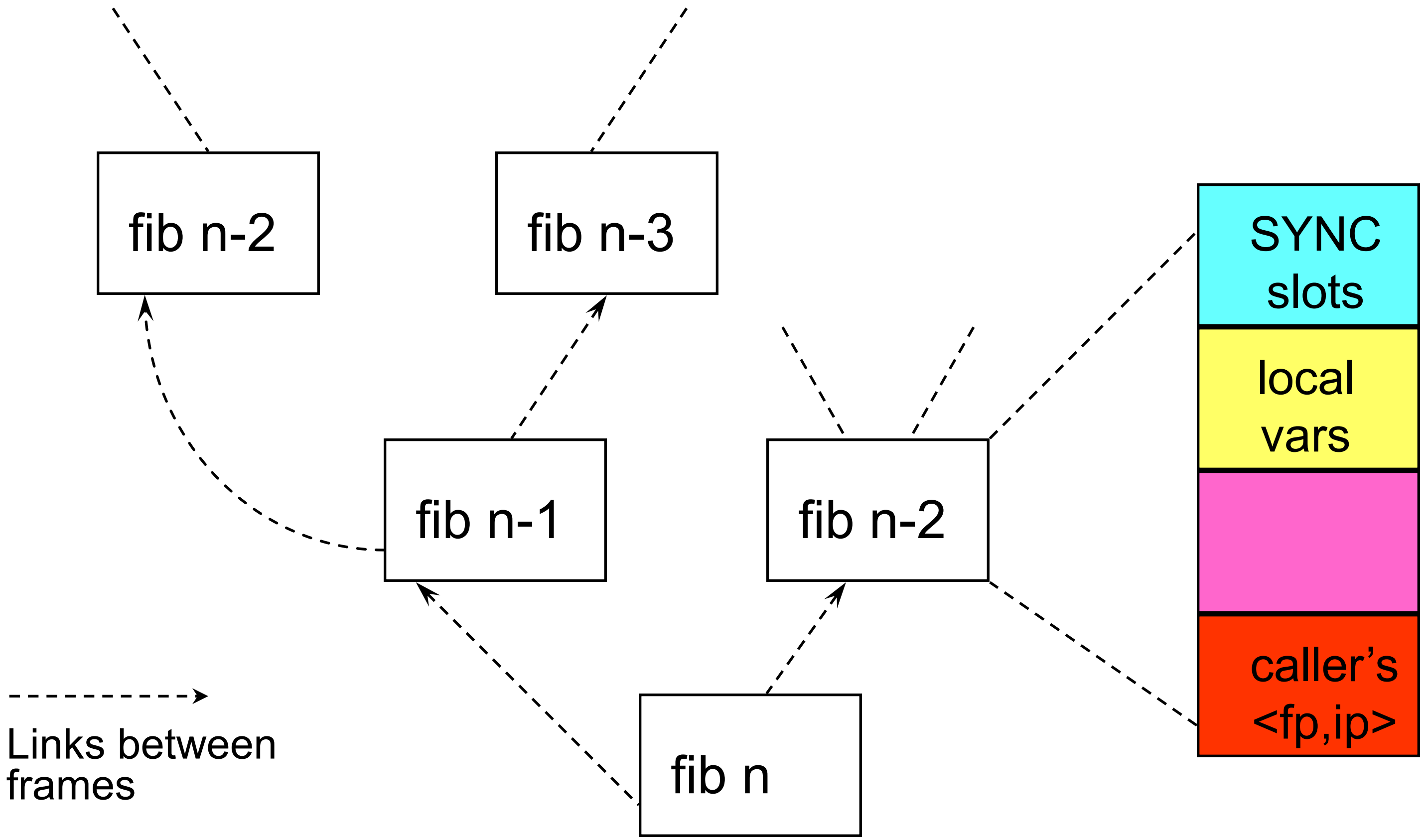
Table 1. EARTH Instruction Set

- **Data Transfer & Synchronization**
  - DATA_SPAWN value, dest_addr, fp, ip
  - DATA_SYNC value, dest_addr, fp, ss_off
  - BLOCKDATA_SPAWN src_addr, dest_addr, size, fp, ip
  - BLOCKDATA_SYNC src_addr, dest_addr, size, fp, ss_off
- **Split_phase Data Requests**
  - GET_SPAWN src_addr, dest_addr, fp, ip
  - GET_SYNC src_addr, dest_addr, fp, ss_off
  - GET_BLOCK_SPAWN src_addr, dest_addr, size, fp, ip
  - GET_BLOCK_SYNC src_addr, dest_addr, size, fp, ip
- **Function Invocation**
  - INVOKE dest_PE, f_name, no_params, params
  - TOKEN f_name, no_params, params
  - END_FUNCTION

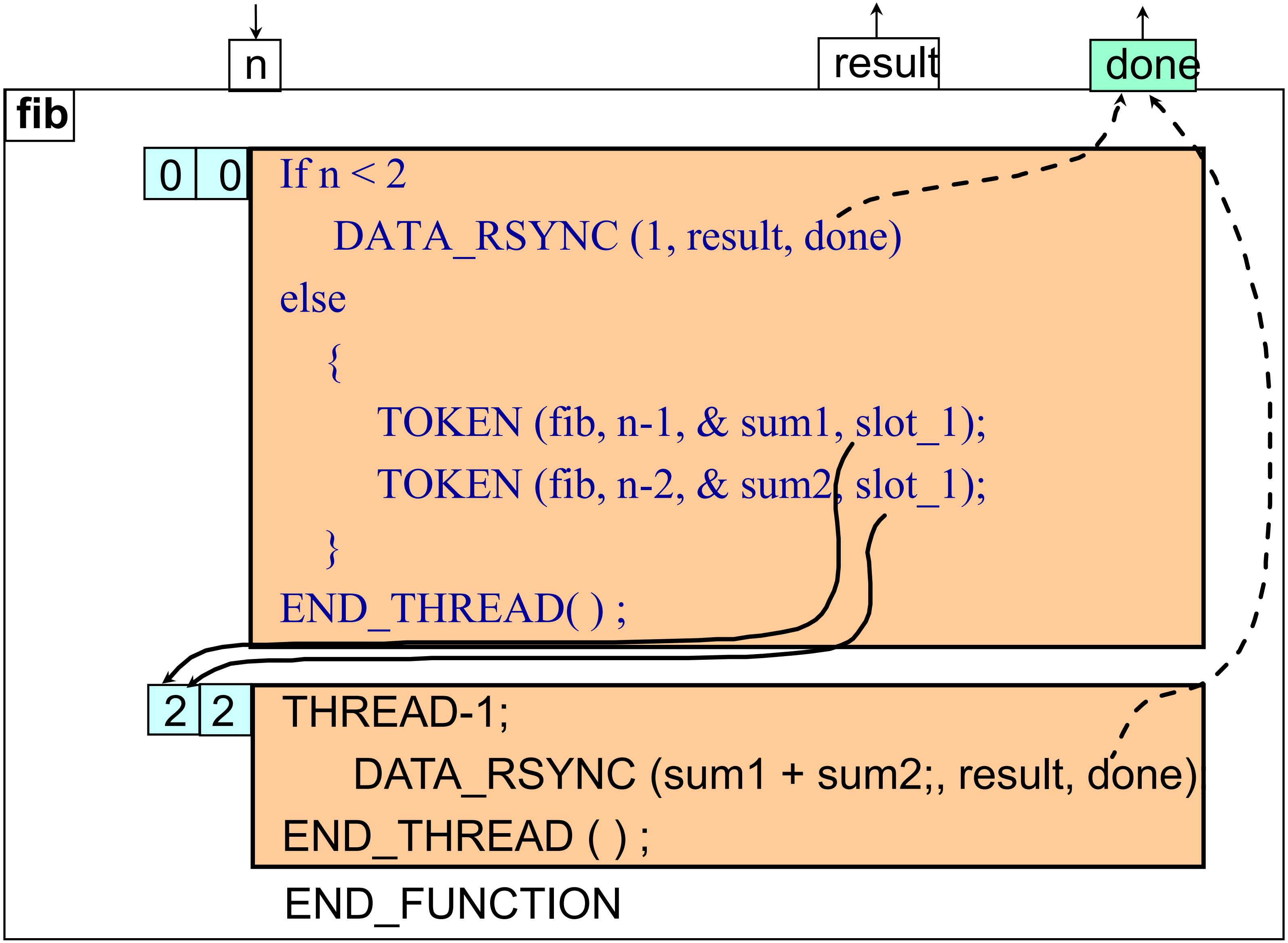
EARTH-MANNA Benchmark Programs

- **Ray Tracing** is a program for rendering 3-D photo-realistic images
- **Protein Folding** is an application that computes all possible folding 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
- **Paraffins** 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.

Parallel Function Invocation



Tree of "Activation Frames"



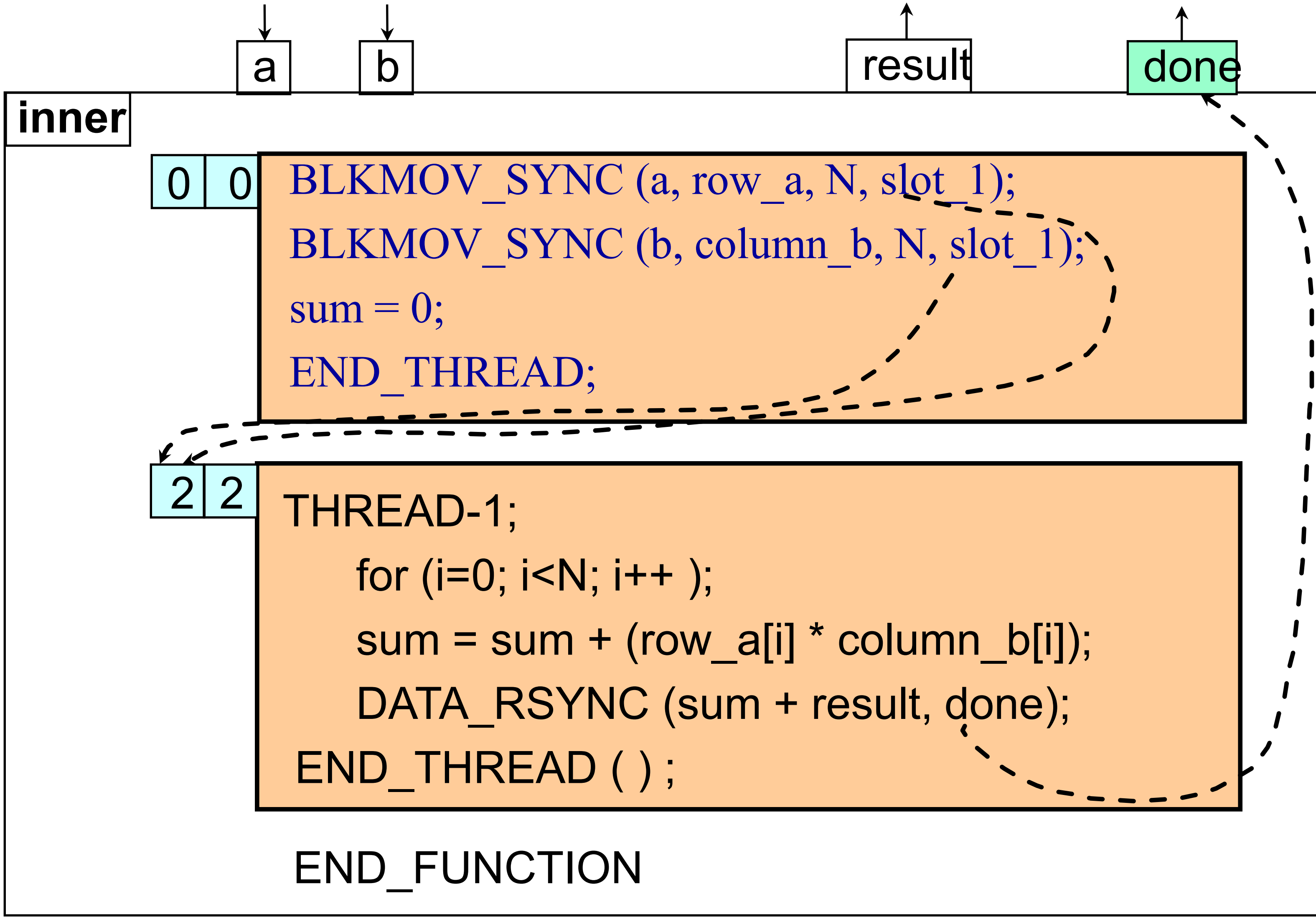
The Fibonacci Example

Matrix Multiplication

```
void main ( )
{
    int      i,  j,  k;
    float  sum;

    for (i=0; i < N;  i++)
        for (j=0; j < N ; j++)   {
            sum = 0;
            for (k=0; k < N;   k++)
                sum = sum + a [i] [k] * b [k] [j]
            c [i] [j] = sum;
        }
}
```

Sequential Version



The Inner Product Example

main

0 | 0

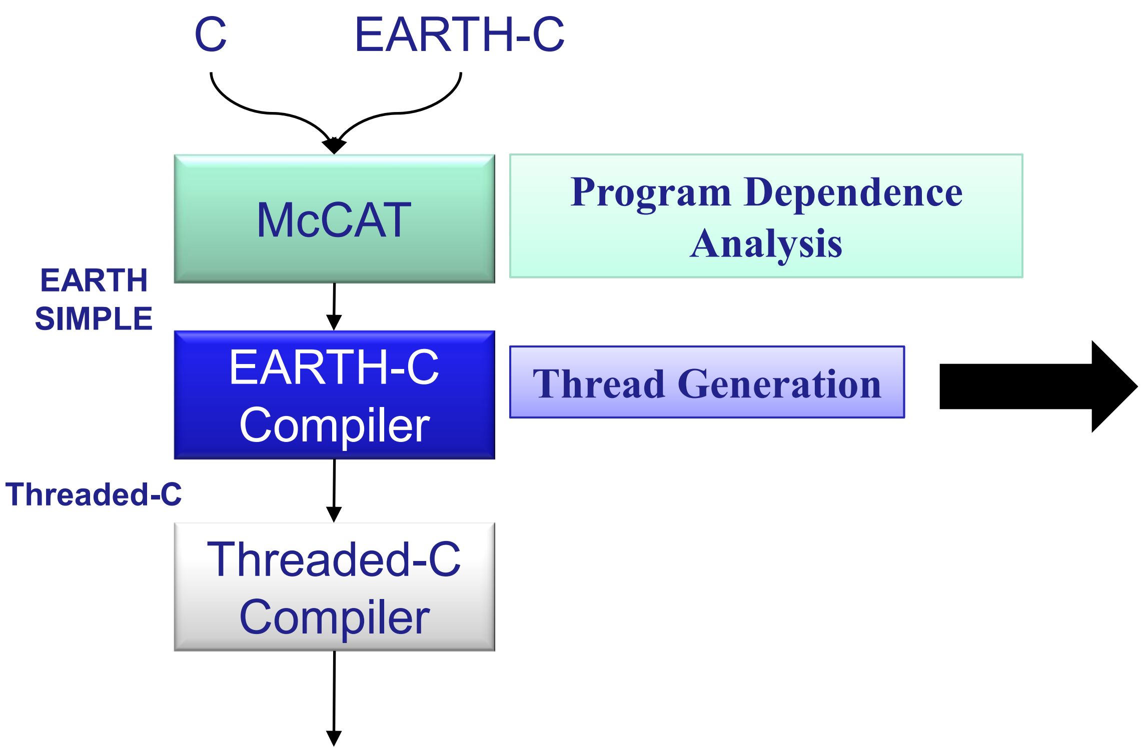
```
for (i=0; i<N; i++)
    for (j=0; j<N; j++)  {
        row_a = a [i];
        column_b = b [j];
        TOKEN (inner, &c[I][j], row_a,
                              column_b,slot_1); }
END_THREAD;
```

N*N | N*N

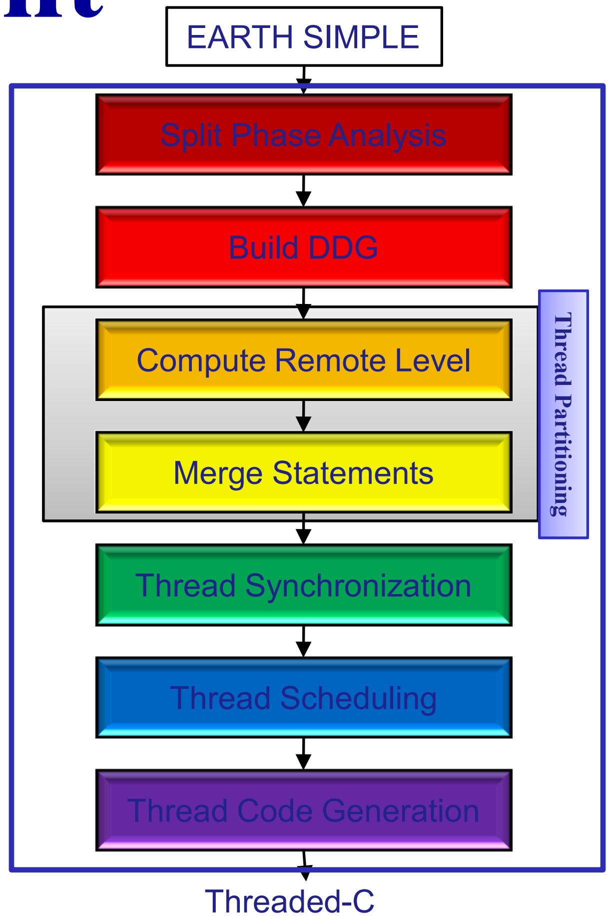
```
THREAD-1;
    RETURN ( );
END- THREAD
```

The Matrix Multiplication Example

EARTH-C Compiler Environment

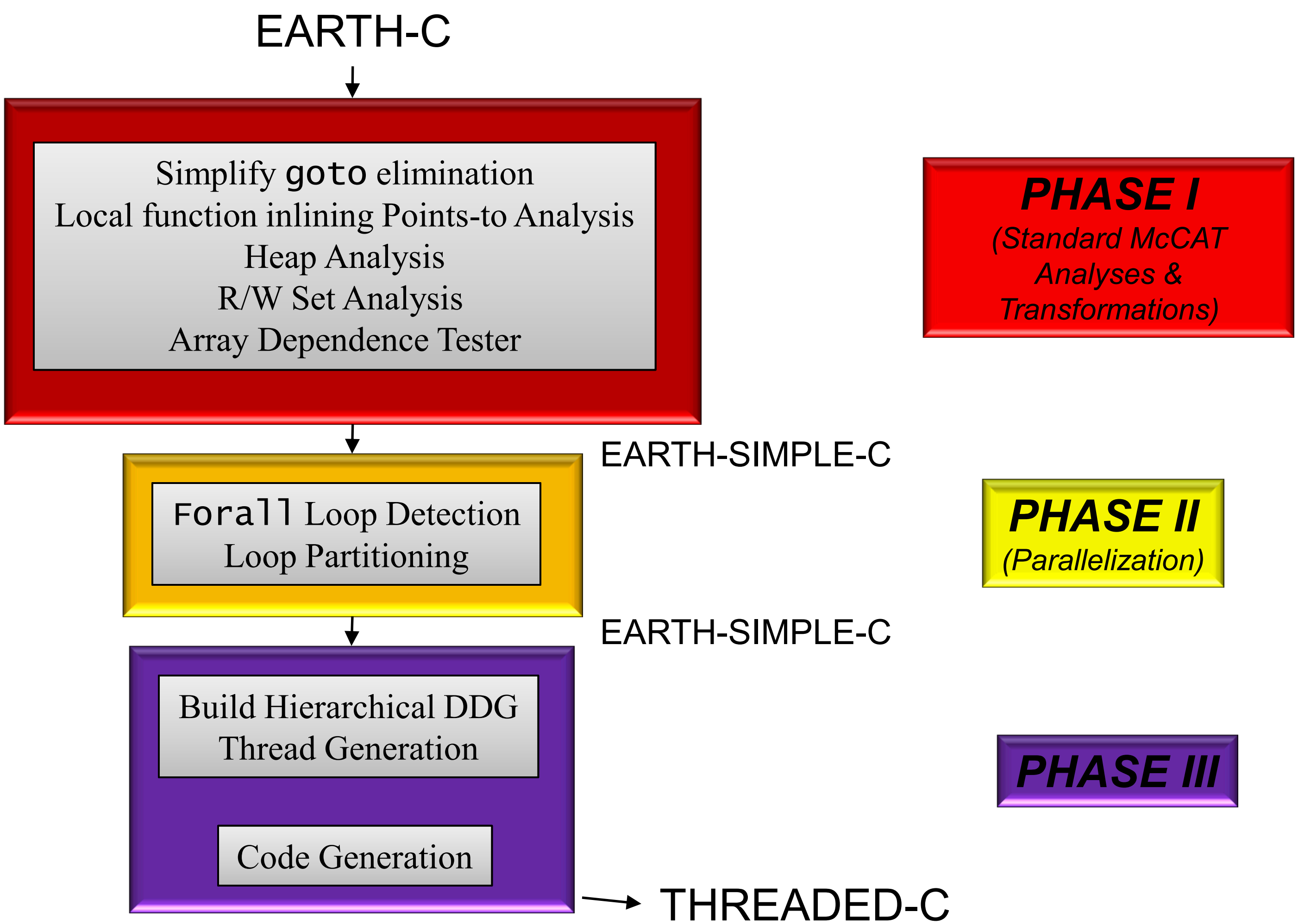


EARTH Compilation Environment



The EARTH Compiler

The McCAT/EARTH Compiler



Advanced Features in Threaded-C Programming

Main Features of EARTH

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

*Features unique to the EARTH model in comparison to the CILK model

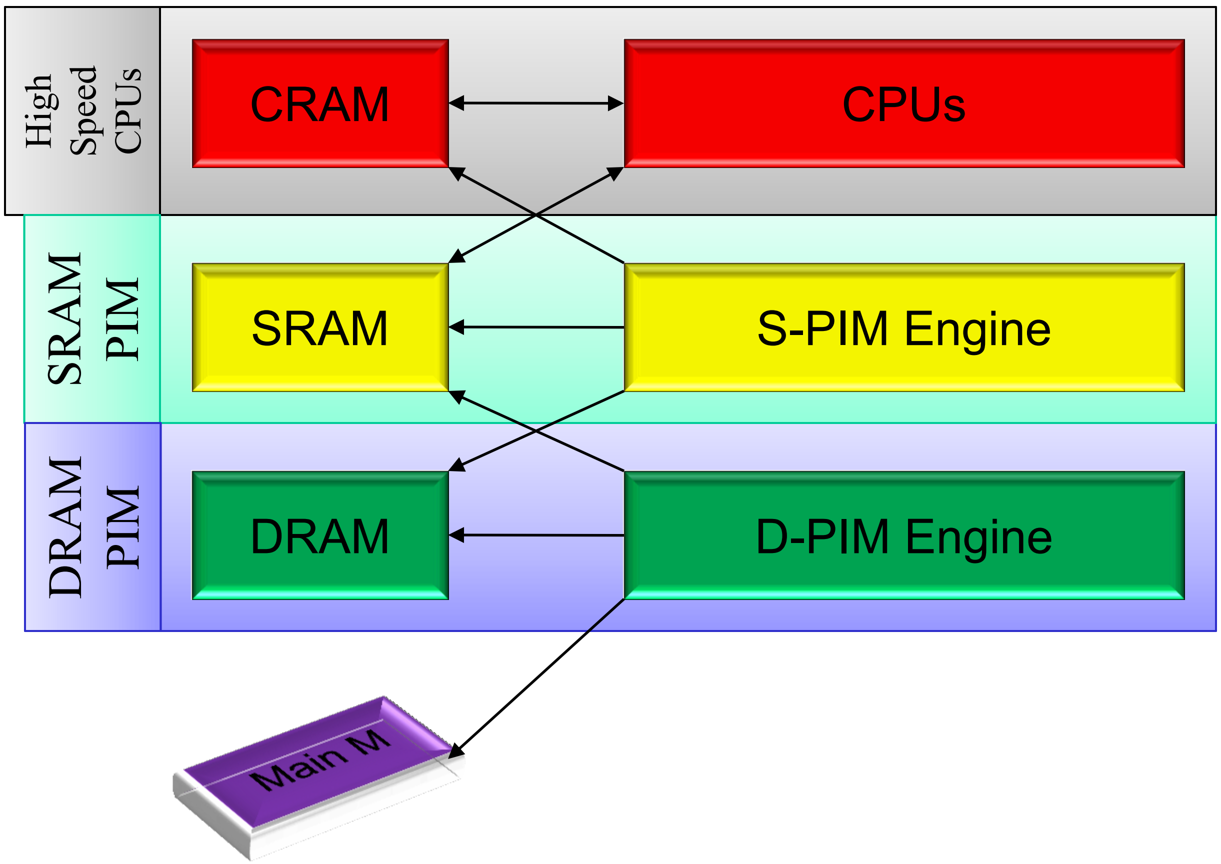
Summary of EARTH-C Extensions

- Explicit Parallelism
  - Parallel versus Sequential statement sequences
  - `Forall` loops
- Locality Annotation
  - Local versus Remote Memory references (global, local, replicate, …)
- Dynamic Load Balancing
  - Basic versus remote function and invocation sites

Percolation Model

under the DARPA HTMT Architecture Project

A User's Perspective



Primary Execution Engine

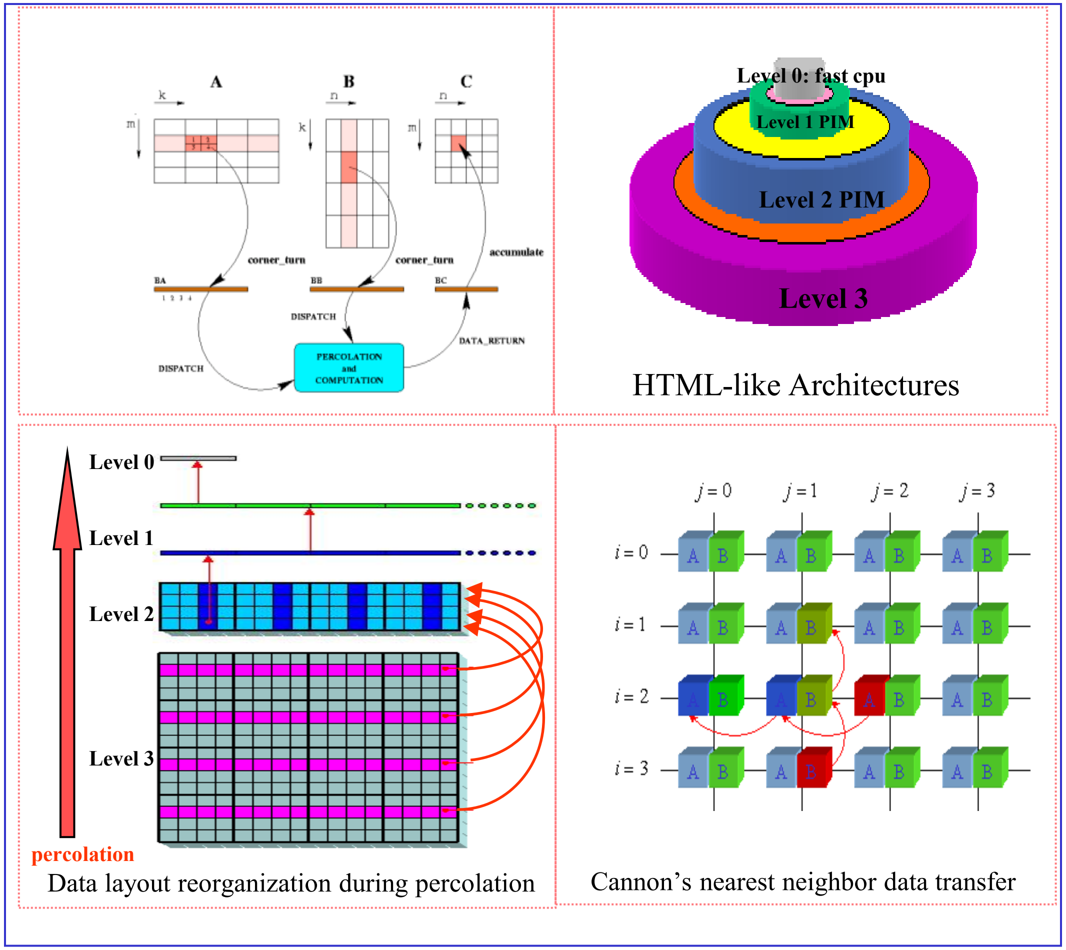
Prepare and percolate "parceled threads"

Perform intelligent memory operations

Global Memory Management

The Percolation Model

- **What is percolation?**
  dynamic, adaptive computation/data movement, migration, transformation in-place or on-the fly to keep system resource usefully busy
- **Features of percolation**
  - both data and thread may percolate
  - computation reorganization and data layout reorganization
  - asynchronous invocation



An Example of percolation—Cannon’s Algorithm

Another View: Codelets

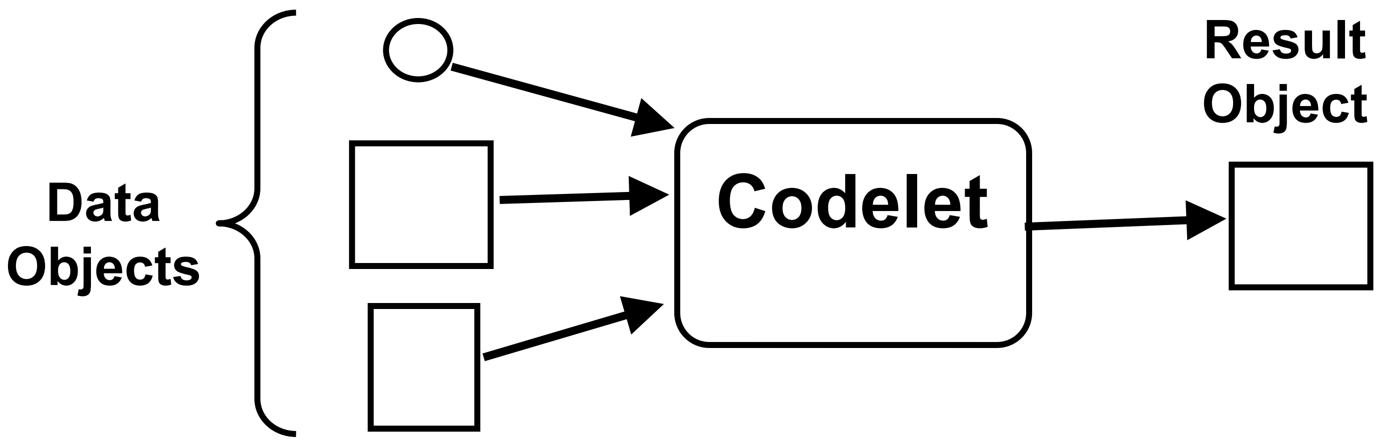
1993: EARTH and 1997: HTMT
Gao, Hum, Theobald

(courtesy: Jack Dennis, DF Workshop, Oct 10. 2011, Gavelston, Tx)

- Group Instructions and Data into Blocks
- Pre-Fetch Input Data
- Non-Pre-emptive Execution
- Store Results in Fresh Memory
- Completion Enables Successor Codelets
- Requires Dynamic Memory Management

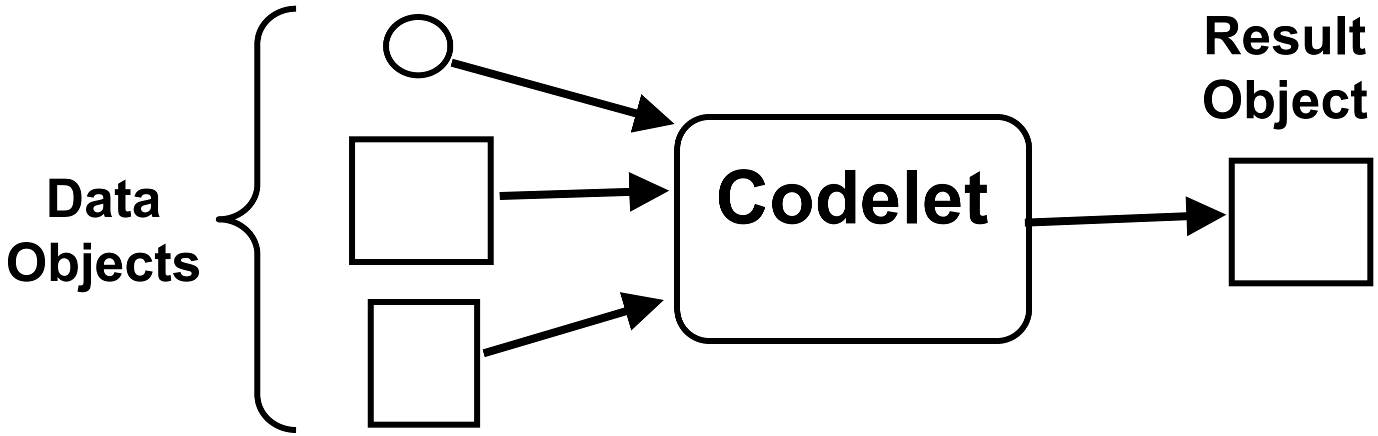
Several Current Projects are Studying Variations on this Concept

The Codelet: A Fine-Grain Piece of Computing



Supports Massively Parallel Computation!

The Codelet: A Fine-Grain Piece of Computing



This Looks Like Data Flow!!