

Programming Models and Storage System for High Performance Computation

Jack Dennis

MIT CSAIL

Guang R Gao

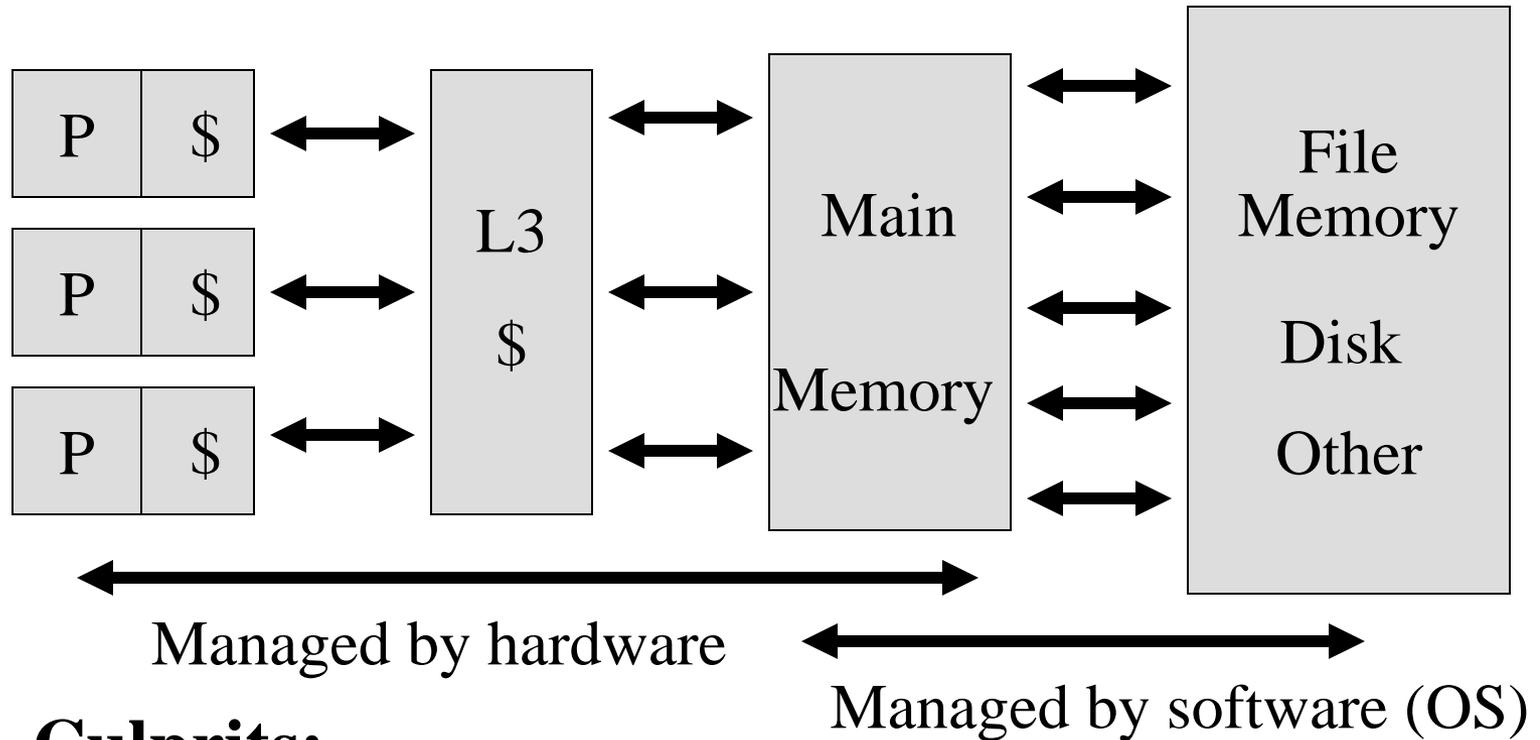
University of Delaware

Vivek Sarkar

Rice University



Problem: I/O Performance



Culprits:

- **Operating System Overhead/Noise**
- **Large Units of Data Transfer**
- **Few Concurrent Transfers (OS Limits)**

Why do we use Software?

Why is the OS such a burden?

Hardware addressing of memory is inadequate:

- Does not support Program Modularity
 - Threads/Jobs run in multiple address spaces – costly communication and sharing
 - Moving objects between memory and file system requires explicit action
- Does not support Security and Privacy
 - All or nothing protection
 - Software required to implement Principle of Least Authority

We can do better!

What is Needed

1. Globally Valid Names for Objects

- To communicate without requiring name translation
- Something more efficient than directory path names.

2. Support for Isolation of Objects/Collections

- Ability to transfer access authority without granting too much

3. Flexible Allocation of Resources

- Provide efficient allocation of processing capacity.
- Extend Virtual memory to encompass all storage

Ideas to Build on

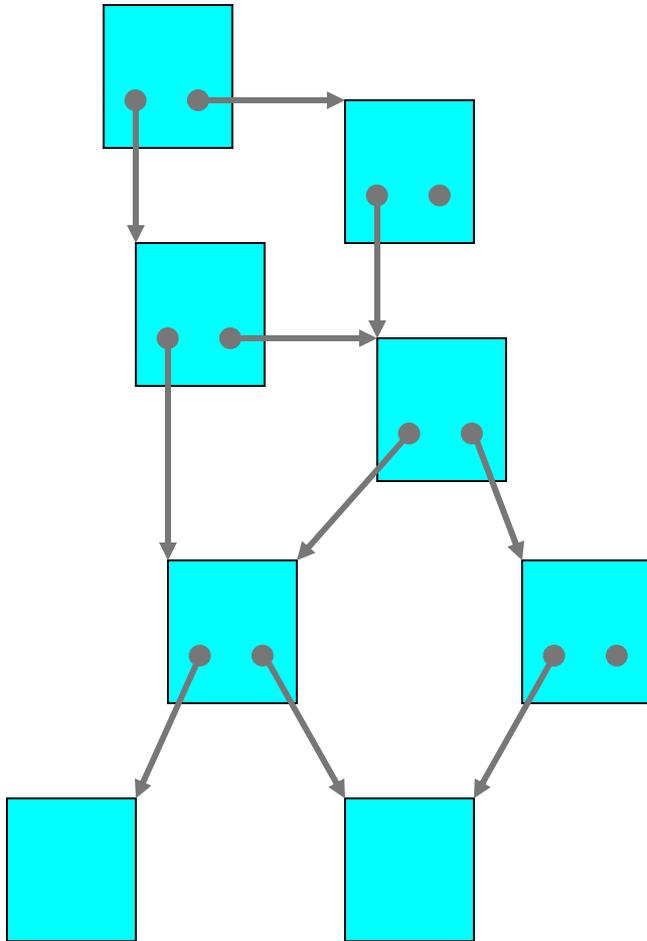
- Global Unique Identifiers of Objects
- Integrate File System with Memory
- Memory Model Based on Tree Structures
- Capability-Based Security
- Functional Programming
- Determinate Execution when Desired
- Principles of Software Modularity
- System-Wide Memory Management (GC)
- Uniform System-Wide Scheduling

The Fresh Breeze Memory Model

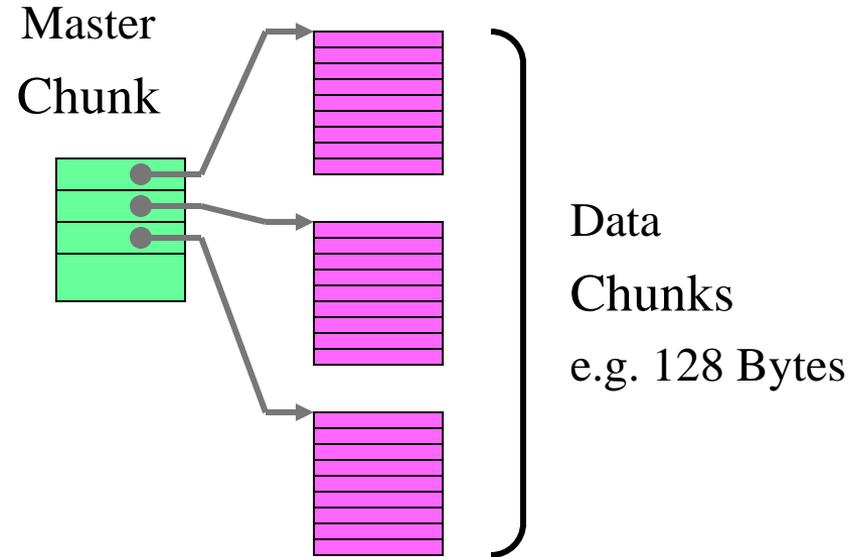
- **Chunk:** A fixed-size unit of memory allocation. On the order of a cache line or VM page.
- Each chunk has a unique identifier with system-wide scope; this is the **pointer** or **UID** of the chunk.
- Chunks may be created, accessed, and released.
- Objects are trees of chunks; unbounded extent.
- No update. Chunks are **Write-Once**
- The graph of chunk references is **Cycle-Free**.
- Concurrent reference count garbage collection.

Data Structures: The Heap as a DAG

Cycle-Free Heap



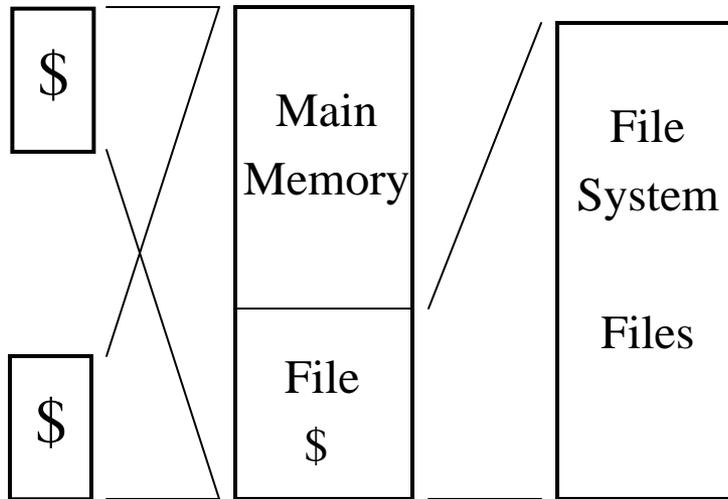
Arrays as Trees of Chunks



- Fan-out as large as 16
- Arrays: Three levels yields 4096 elements (longs)

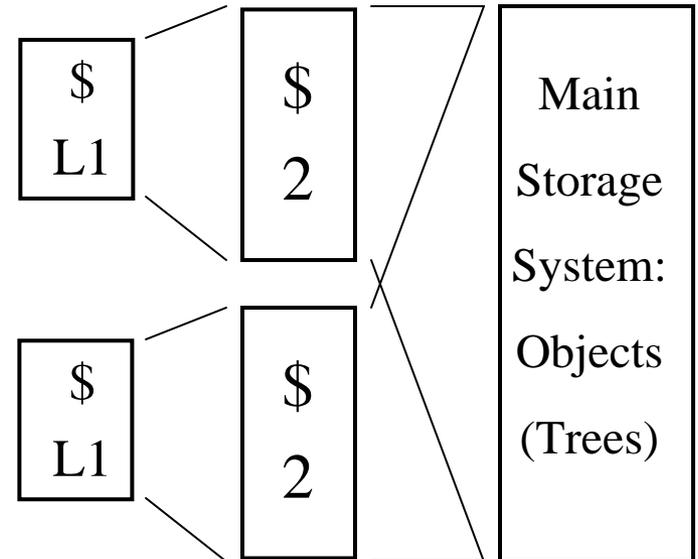
Conventional versus Fresh Breeze

Conventional



- Objects:** Segments in Main Memory (bounded)
Files/Records in the File System
- Naming:** Addresses/pointers in main memory
Path names in file system
- Mapping:** HW assisted in main memory
SW directories in File System
- Consistency:** Cache Protocols, not including
the File System

Fresh Breeze

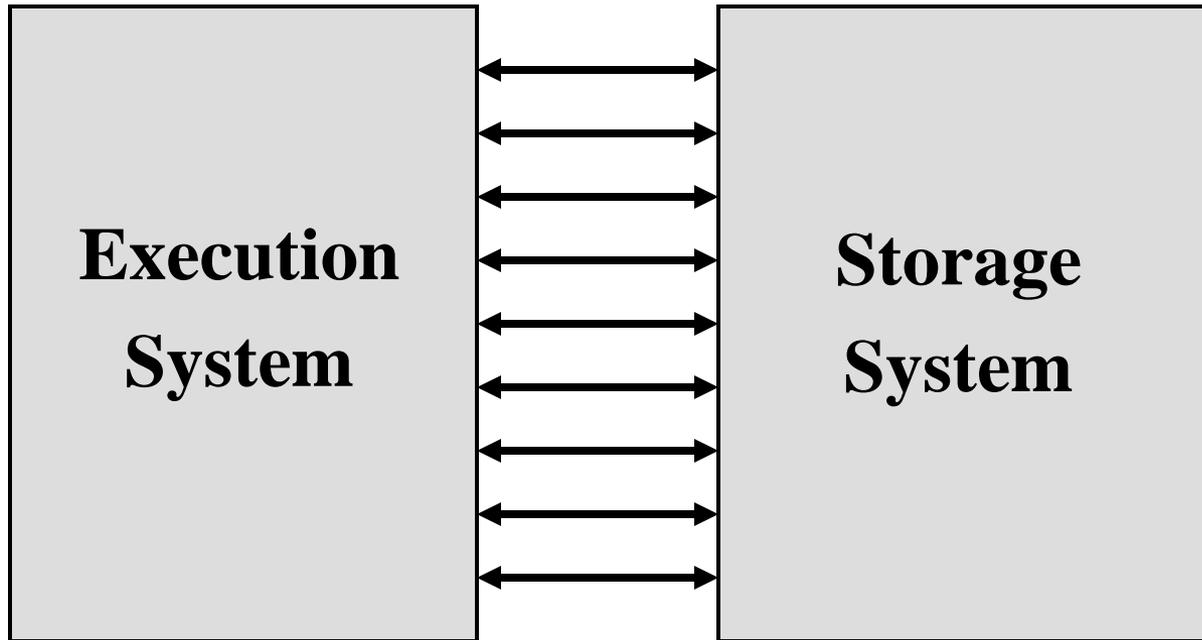


- Objects:** Trees of fixed-size chunks
Uniform at all levels
- Naming:** Global unique identifiers
- Mapping:** Associative HW at all levels
- Consistency:** Write-Once with GC;
No consistency issue

Protection and Security

- A thread must possess a UID to access a chunk
- Given a UID, a process can access the heap subtree reachable from the UID. May be code or data; only code is executable.
- Write protection is not relevant.
- A user gains access to data and code through his/her root directory.

Project Concept



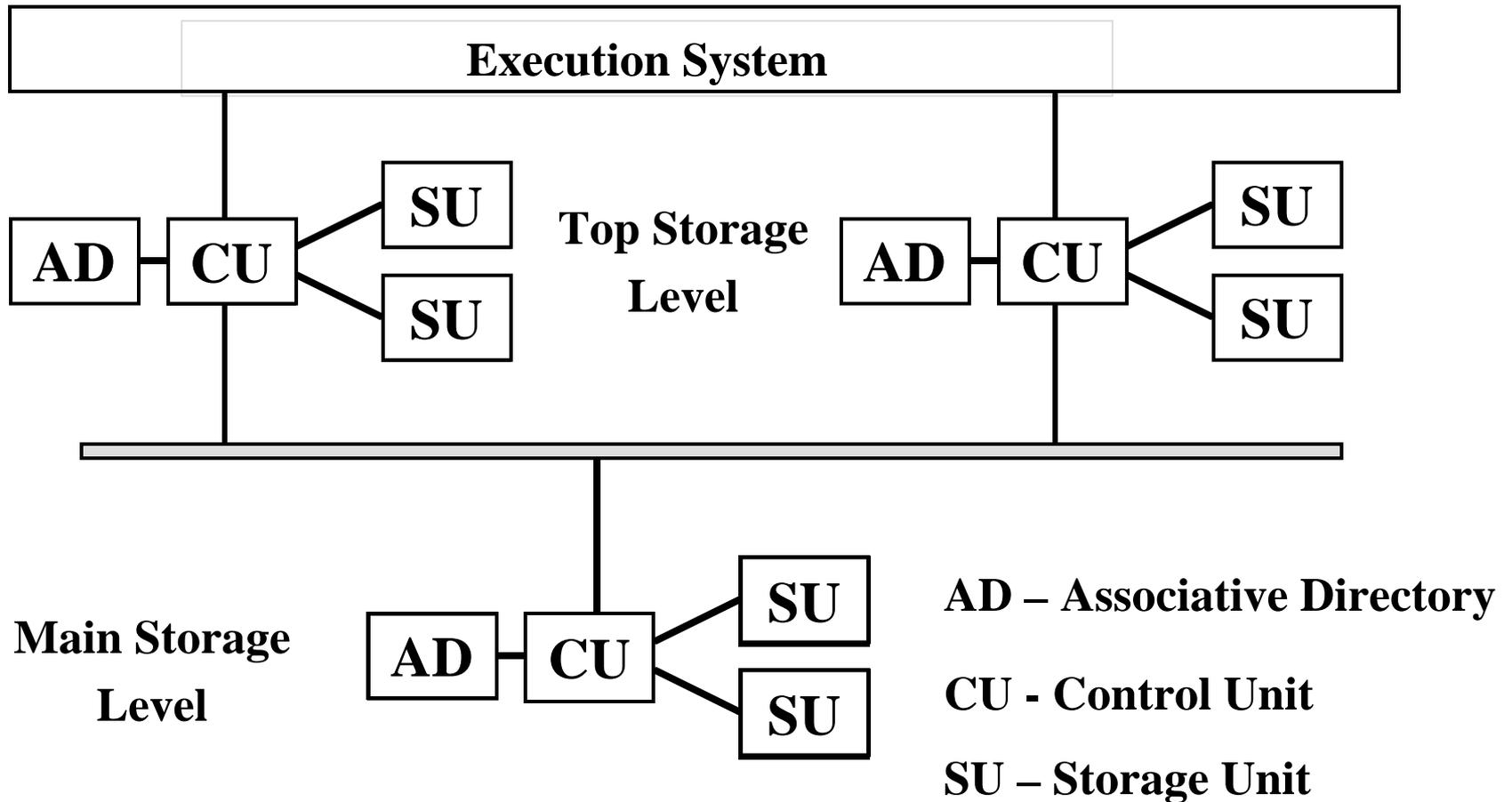
Specified by:

**Programming Models,
Three Versions**

Specified by:

**Fresh Breeze
Memory Model**

Storage System Architecture



The Three Programming Models

1. Declarative Programming (MIT)

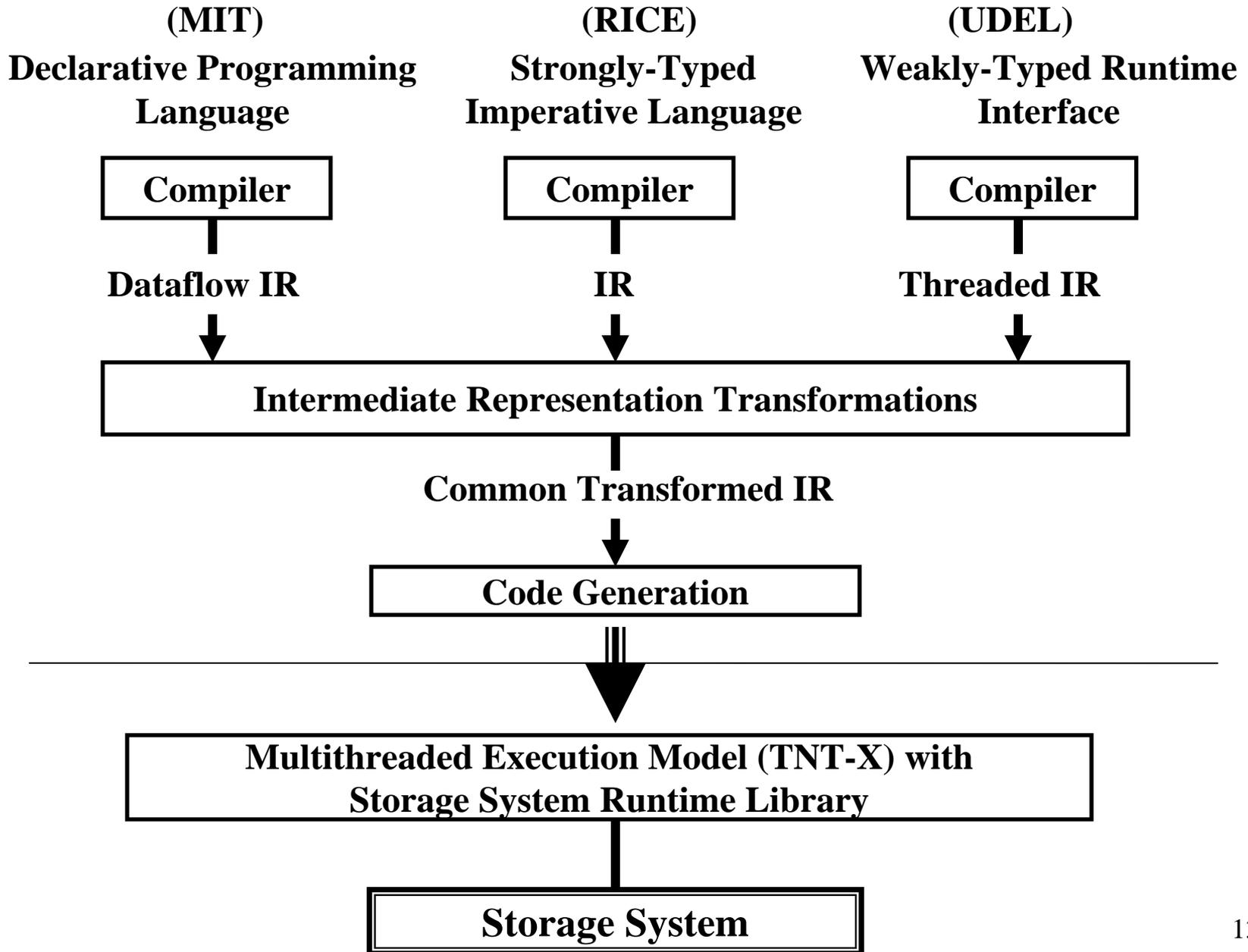
Based on FunJava, a functional dialect of Java

2. Strongly-Typed Imperative Programming (Rice)

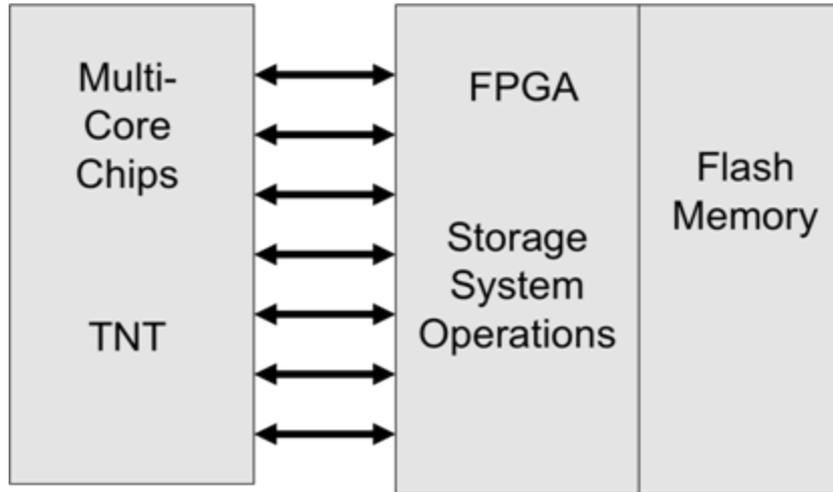
**Based on X10, including the async and finish
Primitives for parallel programming**

3. Weakly-Typed Runtime Interface (API) (UDeI)

**Providing direct access to the TNT API plus
Storage System operations.**



Proposed Experimental System



Evaluation:

Variation of performance with:

- Choice of memory chunk size
- Structure of the Storage System
- Load Balancing and Scheduling policies
- Choice of Programming Model

Candidate benchmark suites:

- **IOR** (<http://sourceforge.net/projects/ior-sio>)
- **IOzone** (<http://www.iozone.org/>)
- **Xdd** (<http://www.ioperformance.com/>)

Candidate simulation platform:

- FAST simulator for IBM Cyclops-64 extended with storage components

Feedback welcome on all these topics

Summary

Programming Models and Storage System for High Performance Computation

- I/O Performance:

The OS is in the way! Addressing means are inadequate!

- New Memory Model

Write once, fixed-size Chunks; Global Unique Identifiers

- Three Programming Models:

Declarative versus Imperative; Strongly versus Weakly Typed Language

- Experimental System:

TiNy Threads + FPGA + Flash

