

First Experiments with an Emulation of the Fresh Breeze Storage System

Jack Dennis

Xiaoxuan Meng

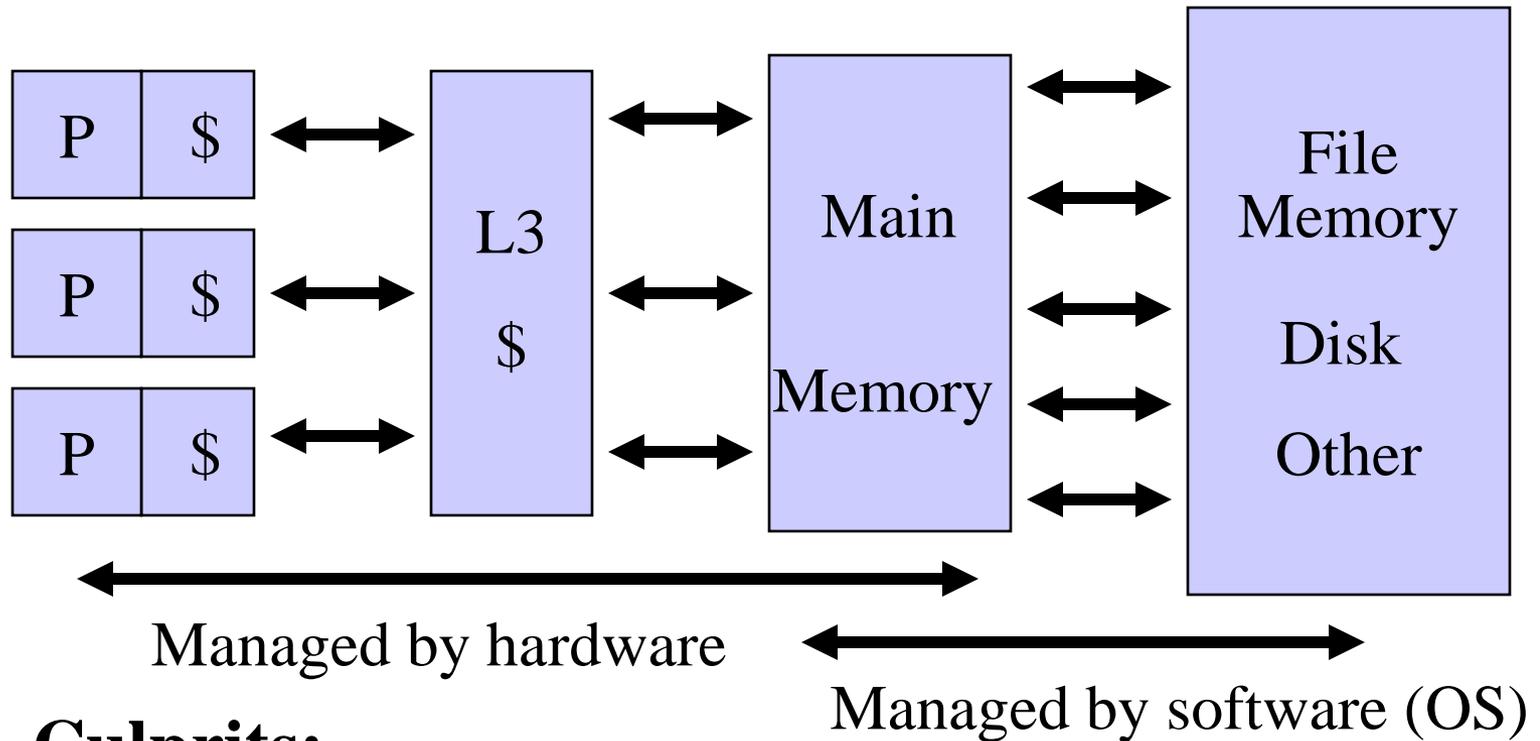
MIT CSAIL

University of Delaware

Funded by NSF HECURA Grant CCF-0937832



Problem: I/O Performance

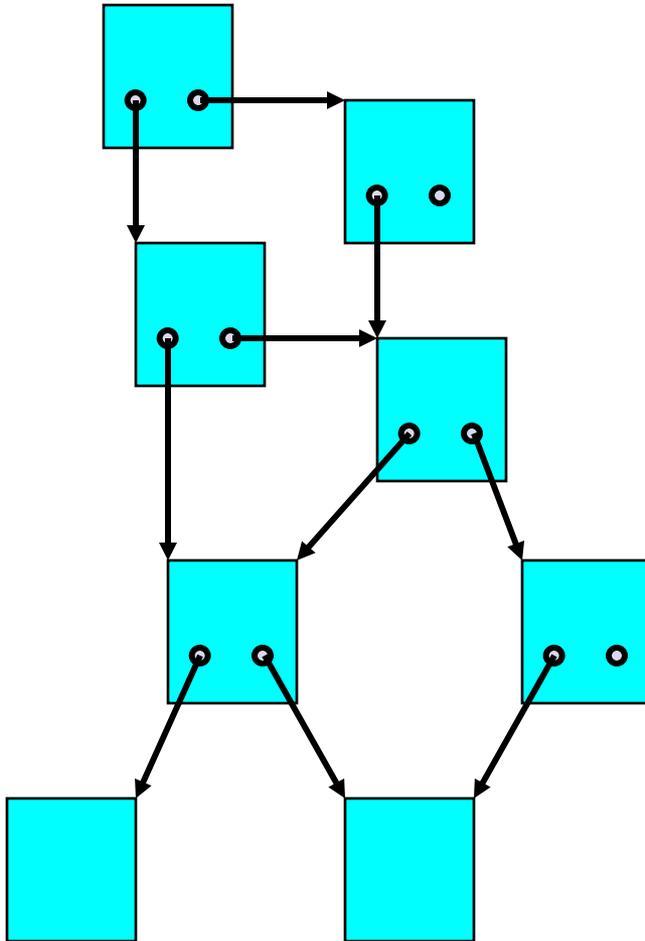


Culprits:

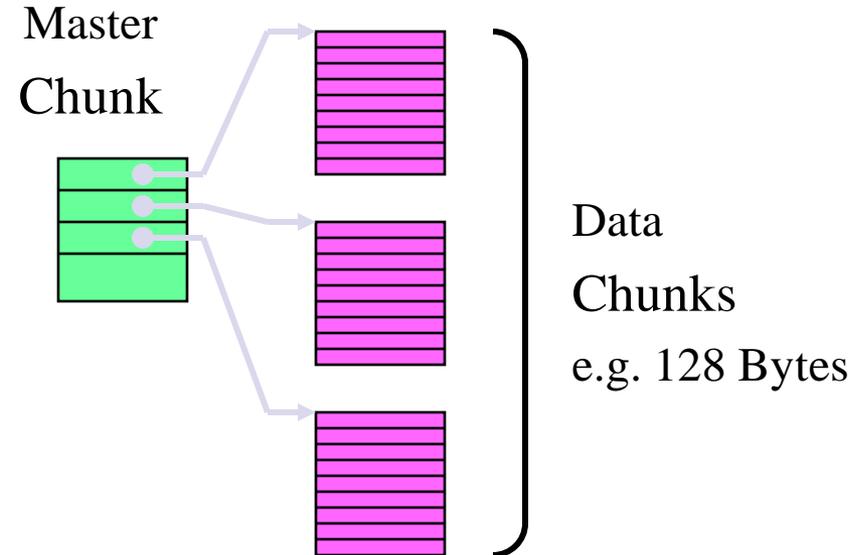
- **Operating System Overhead/Noise**
- **Large Units of Data Transfer**
- **Few Concurrent Transfers (OS Limits)**

The Fresh Breeze Memory Model

Cycle-Free Heap

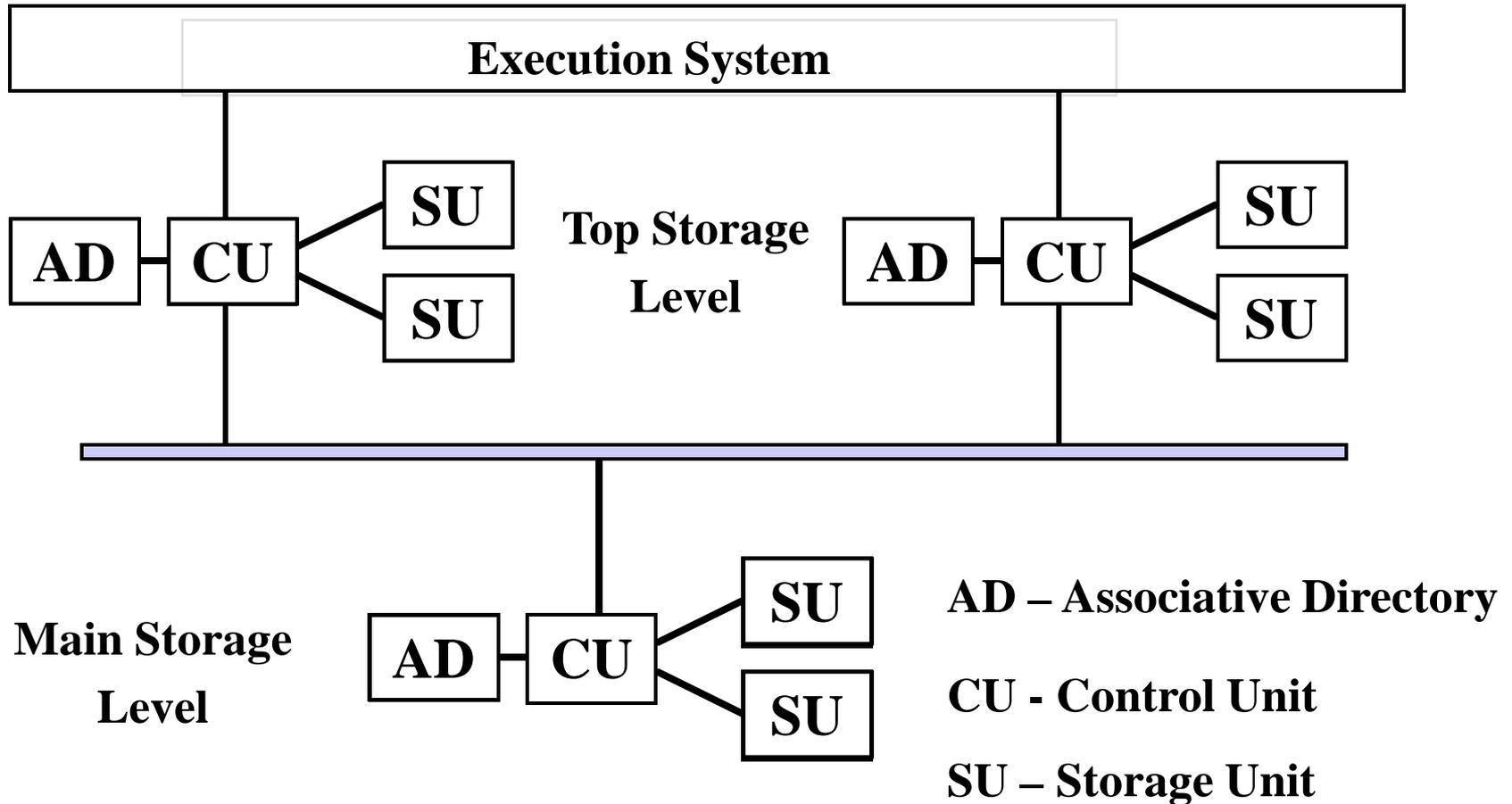


Arrays as Trees of Chunks



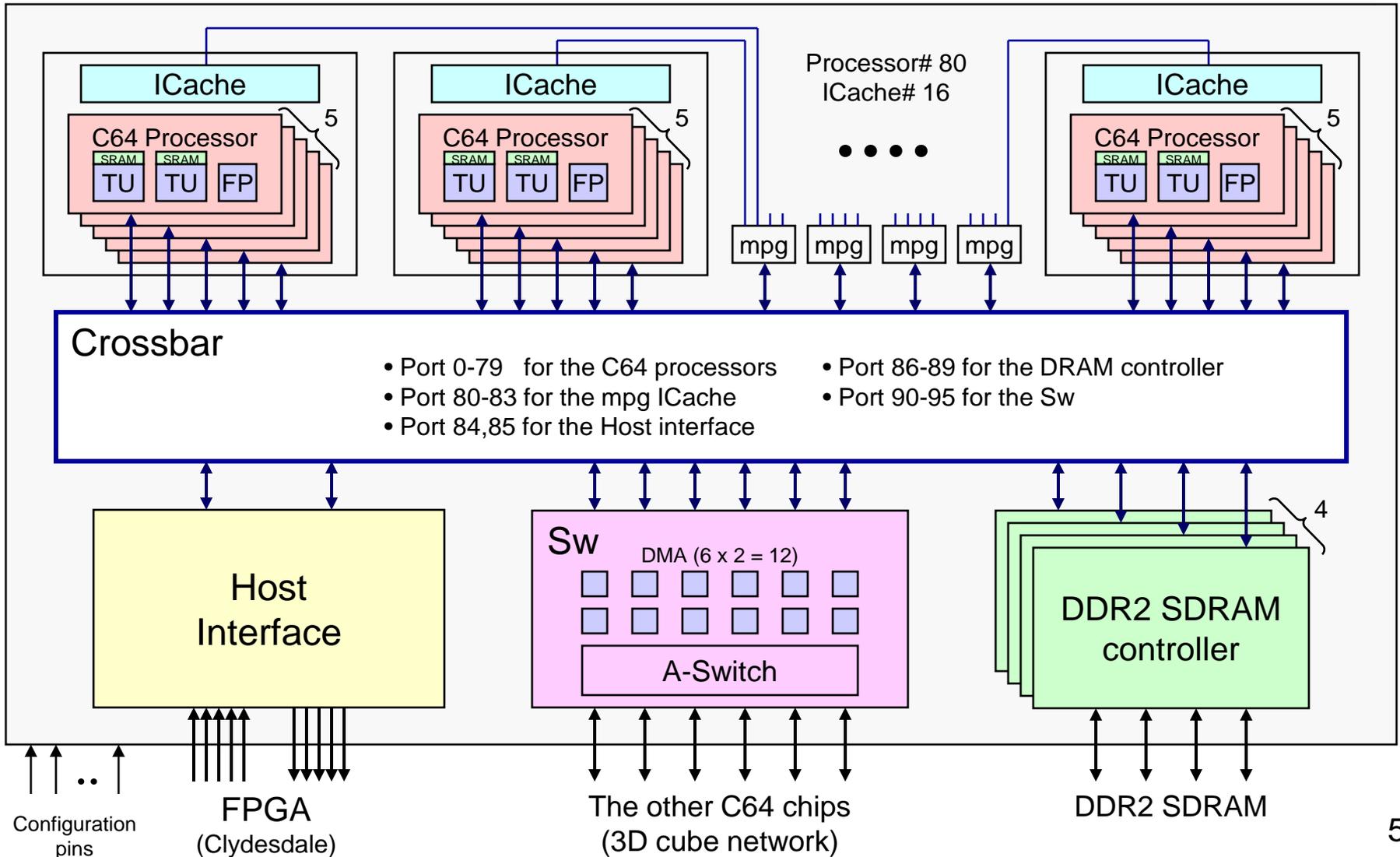
- Fan-out as large as 16
- Arrays: Three levels yields 4096 elements (longs)
- Write-Once then Read Only

Storage System Architecture

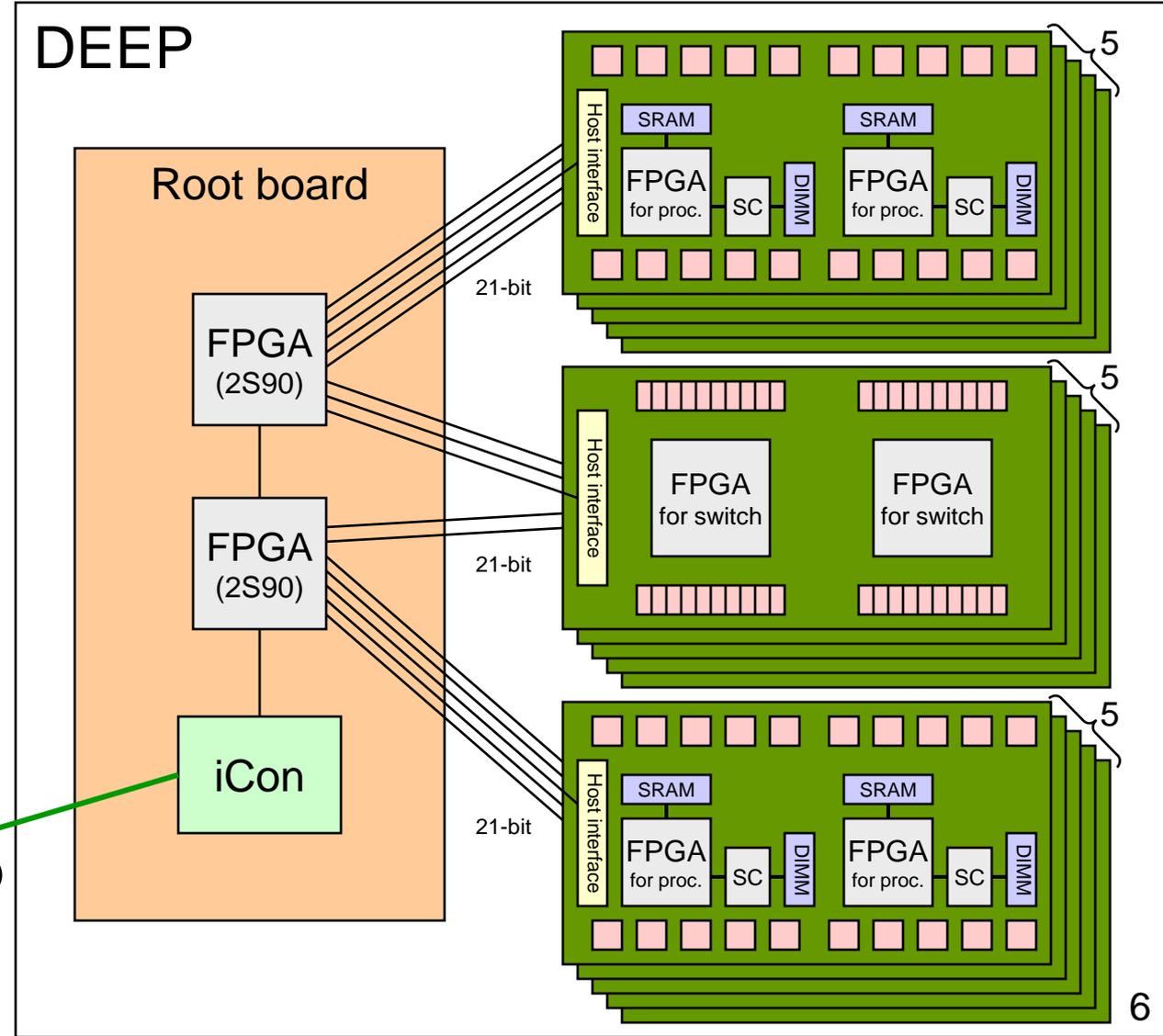
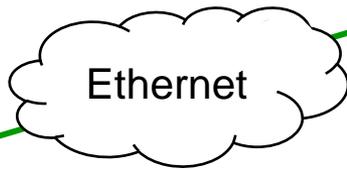
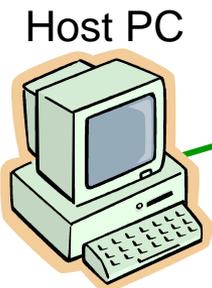


IBM Cyclops 64 chip

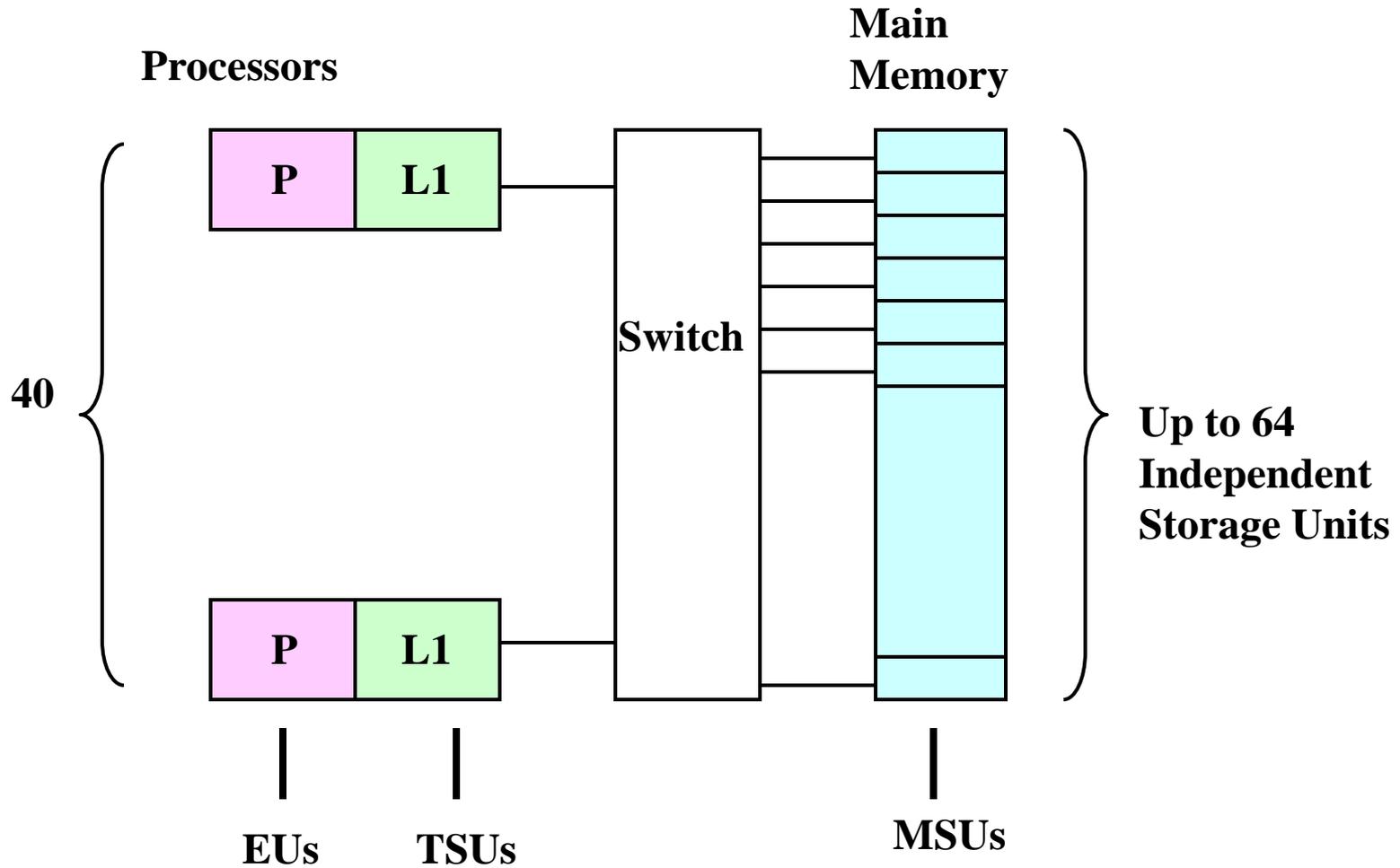
- Each TU has an SRAM (4k x 64-bit = 32kB).
- Each ICache is shared by 5 C64 Processors.
- One Crossbar port is shared by 4 ICaches.



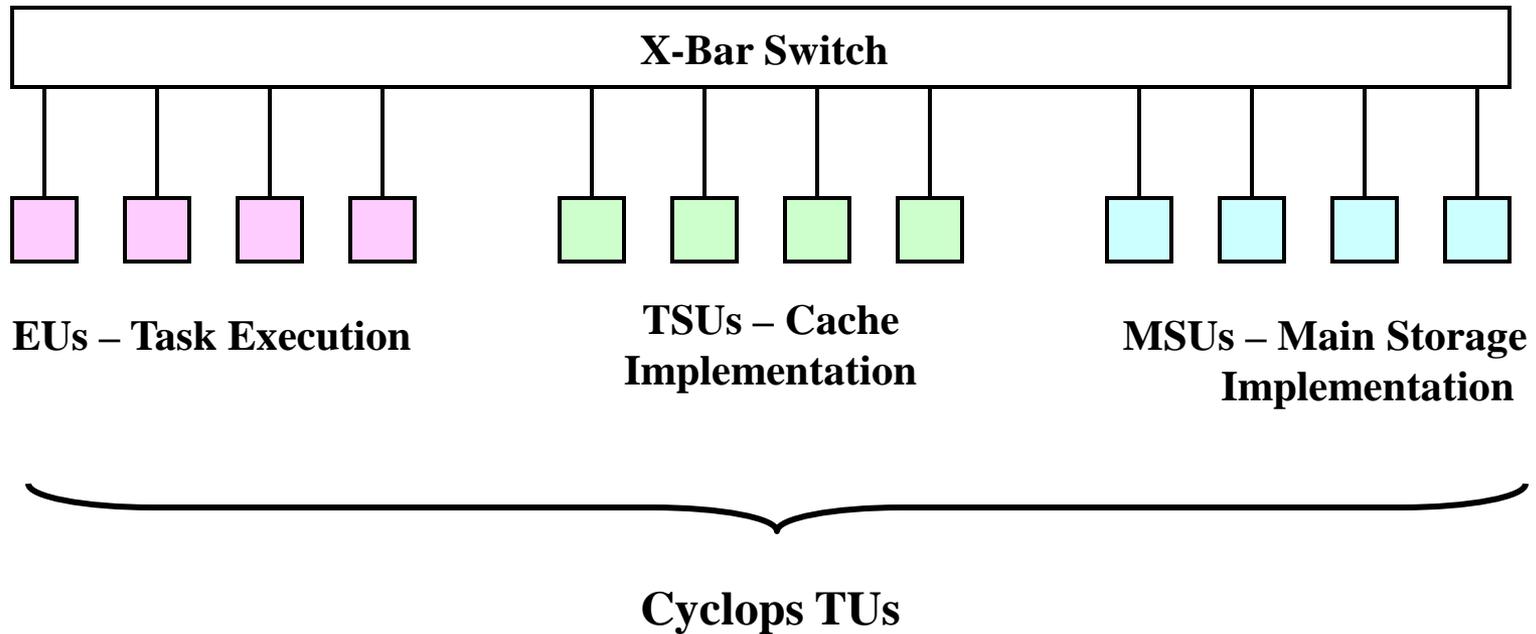
Delaware Enhanced Emulation Platform



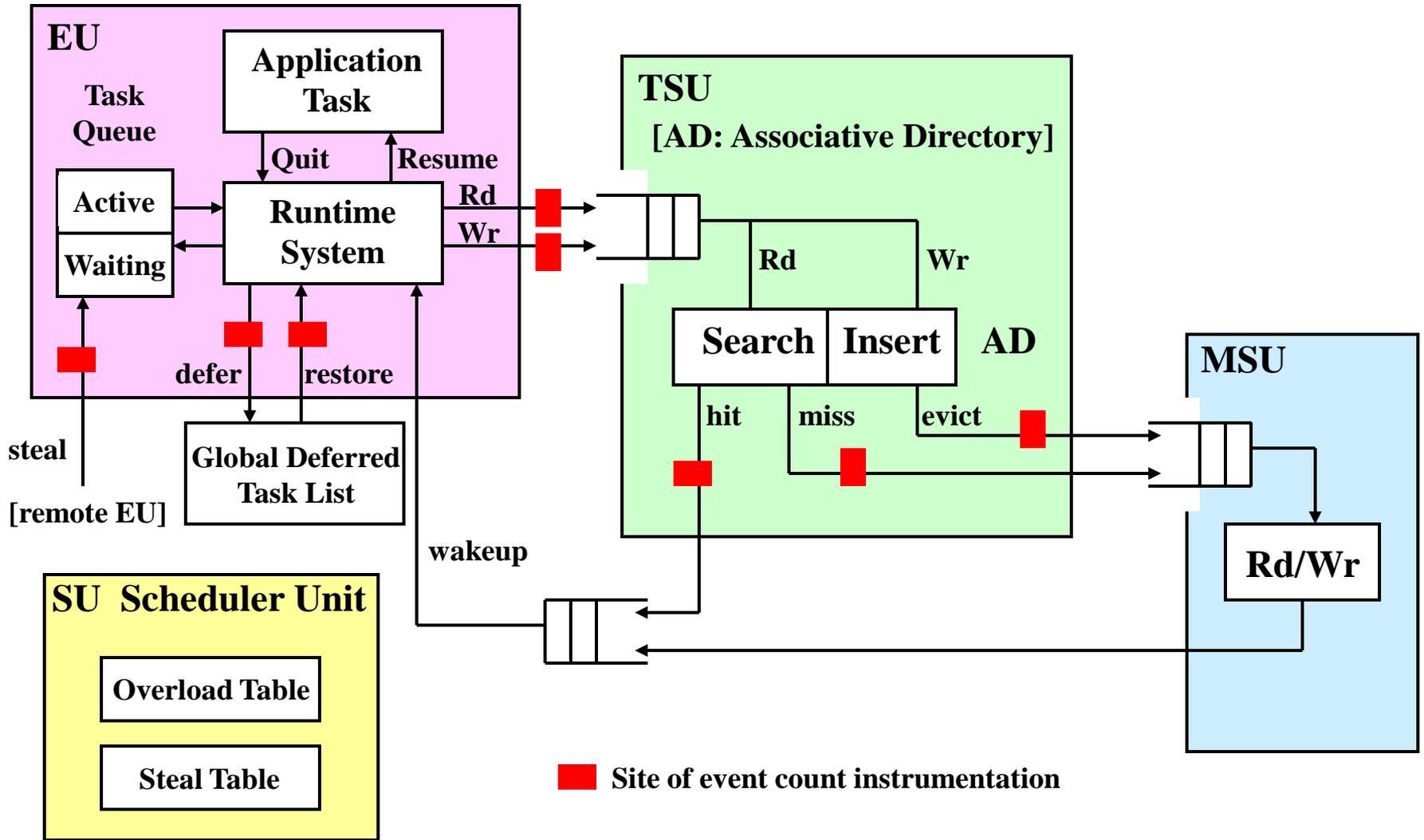
System for Emulation



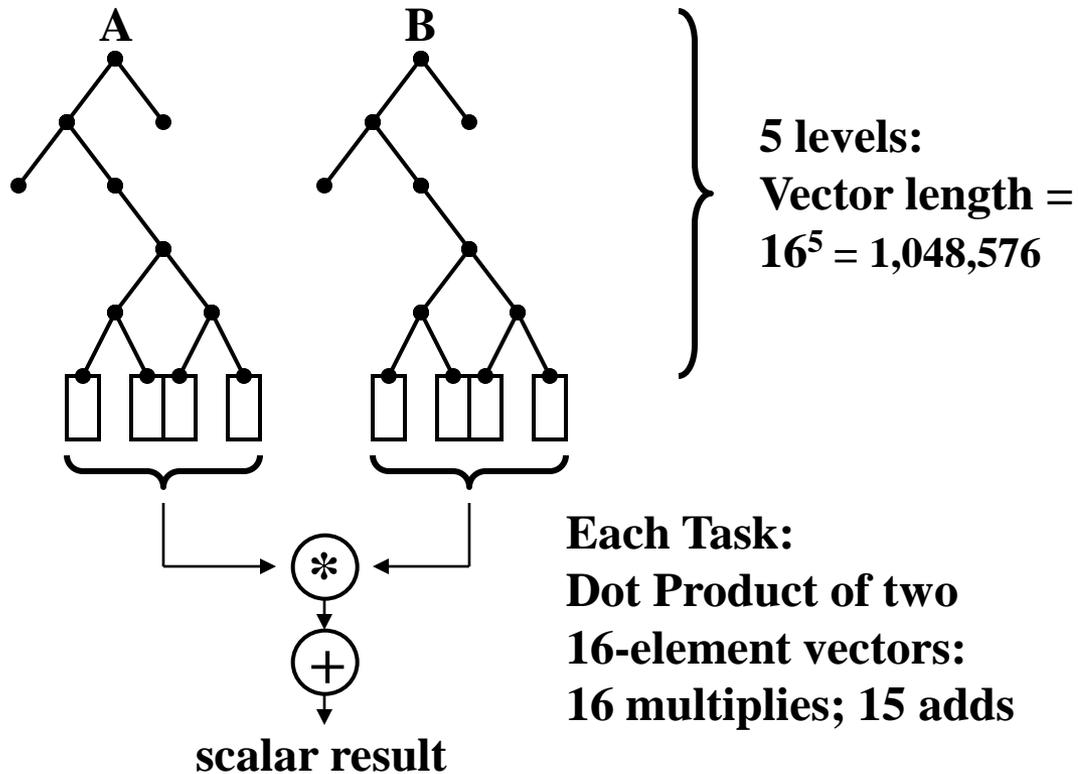
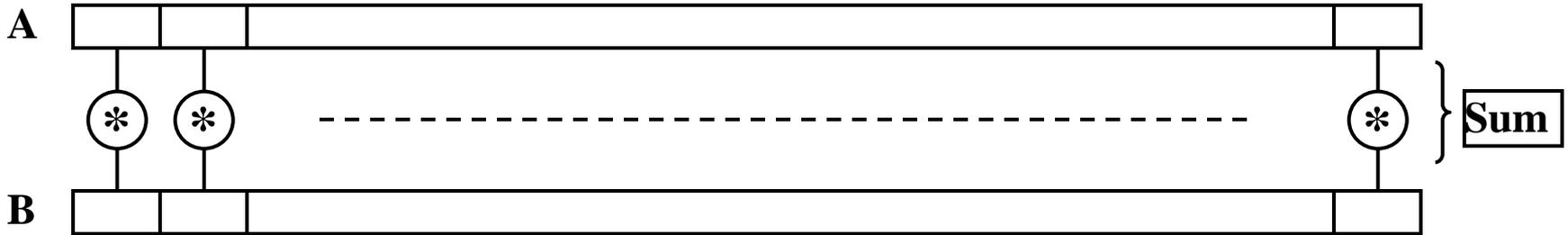
Fresh Breeze Emulation



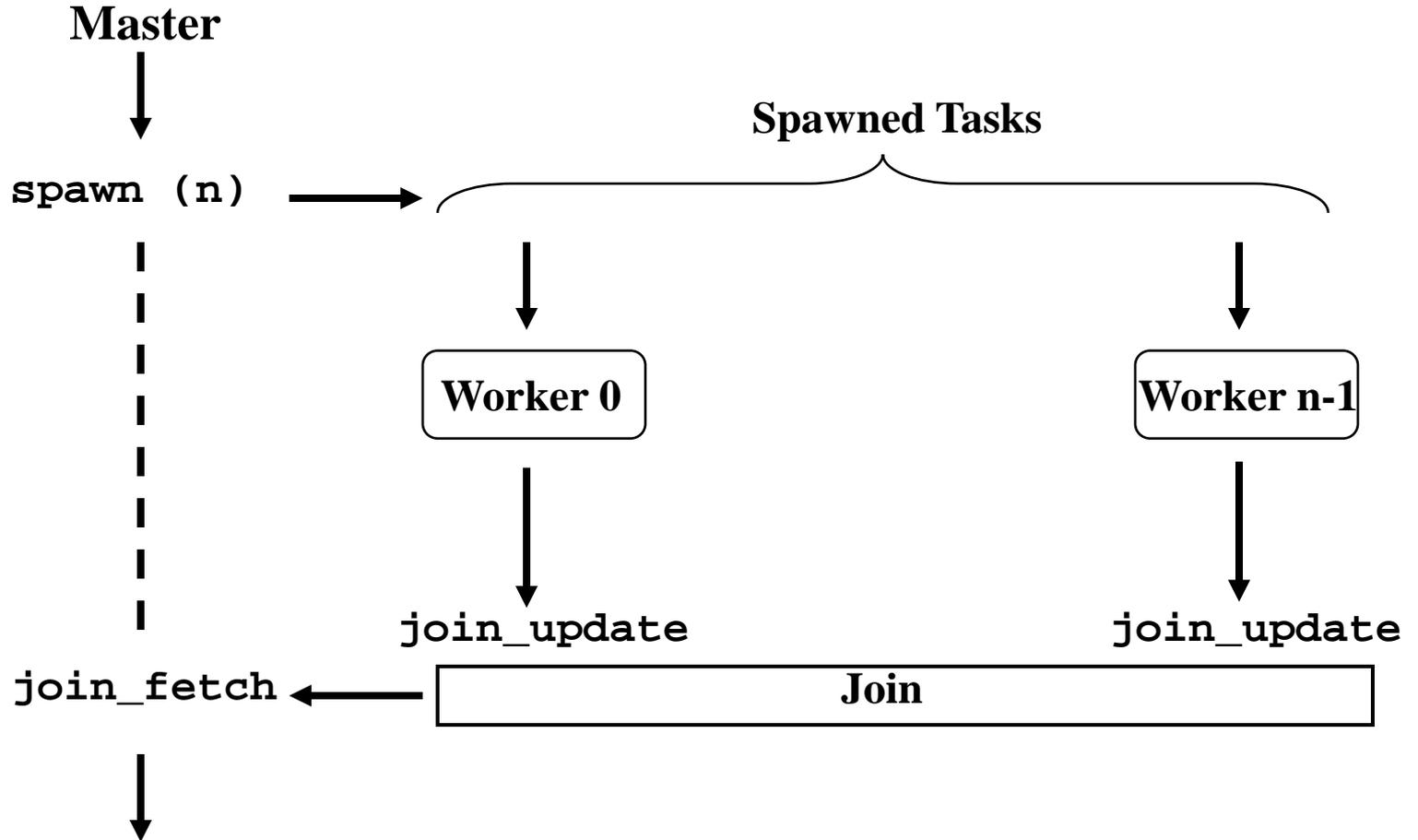
Emulation Scheme



The Dot Product



Spawn and Join

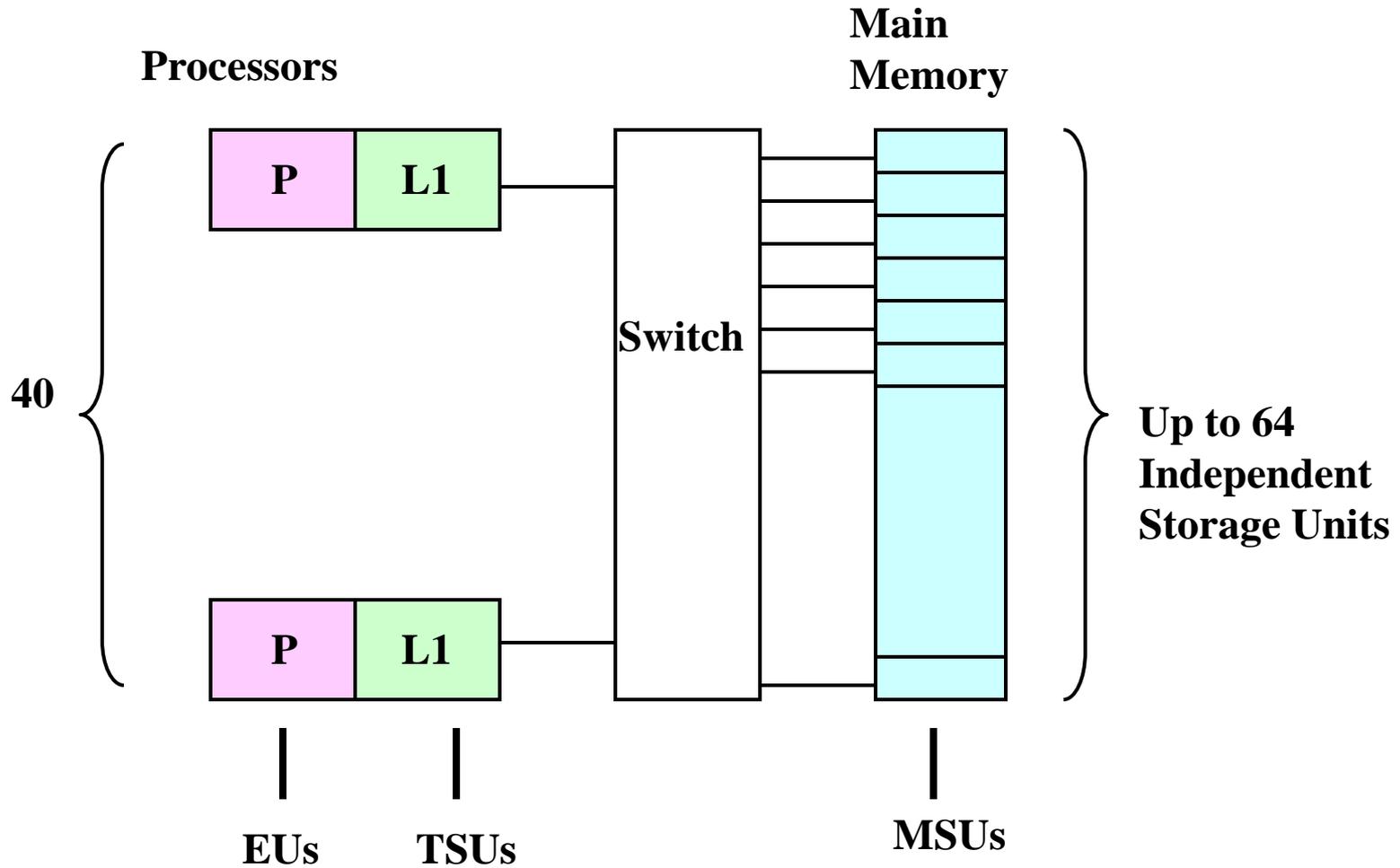


Spawn – Join Programming

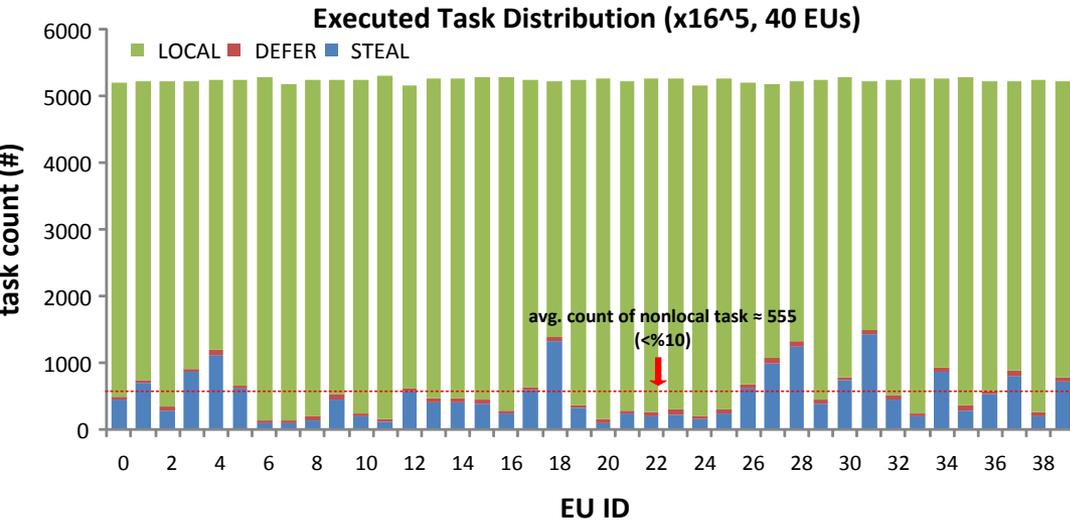
```
void DotProd ( a, b ) {  
    join_init (16, Accumulate ( ) );  
    for ( i = 0; i < 16; i++)  
        spawn_one ( i, DotProd ( a[i], b[i] ) )  
    quit ( );  
}
```

```
void Accumulate ( ) {  
    handle = join_fetch ( );  
    sum = 0;  
    for ( j = 0; j < 16; j++ )  
        sum += handle [j];  
    join_update ( sum );  
}
```

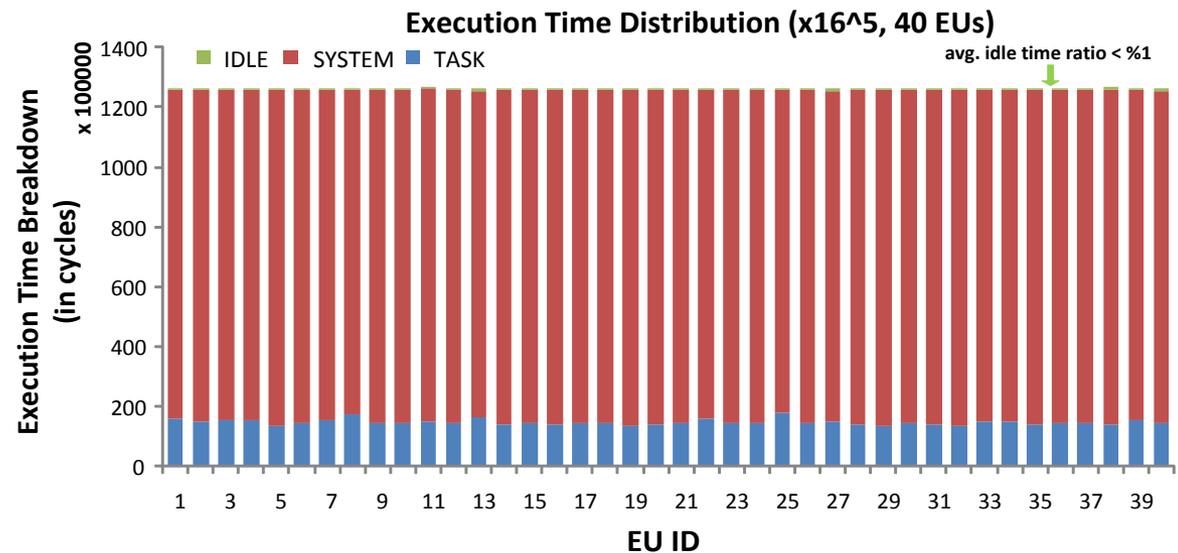
System for Emulation



Task load distribution of parallel dot product program on FB-EXM

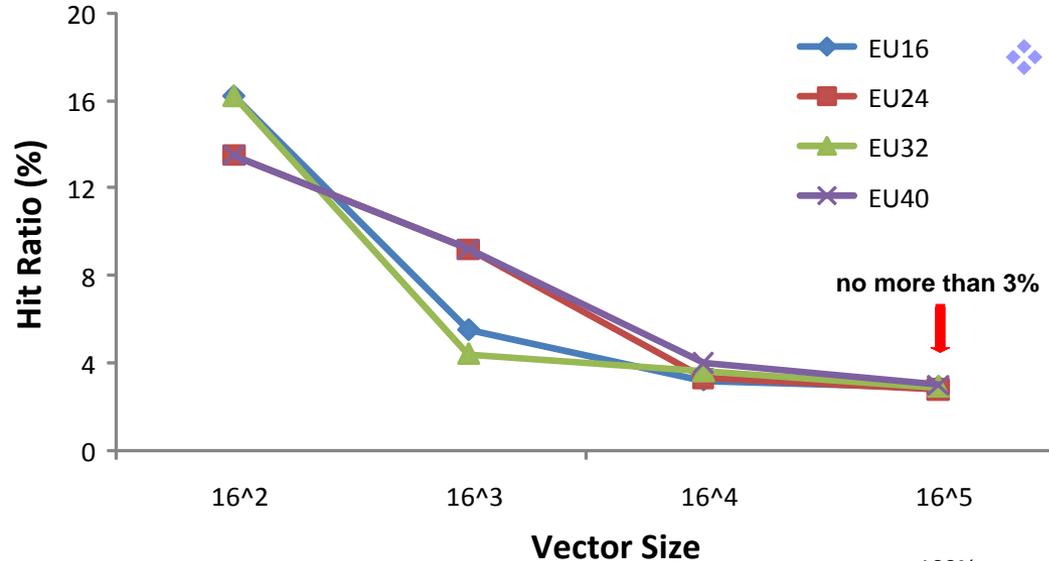


- ❖ Tasks are evenly distributed among EUs by the runtime scheduler
- ❖ Only a small portion of non-local tasks (<10%) exists
 - ❖ DEFER task $\approx 1\%$
 - ❖ STEAL task $\approx 9\%$



- ❖ All the EUs are kept equally busy during execution
 - ❖ TASK time $\approx 12\%$
 - ❖ SYS time $\approx 87\%$
- ❖ The IDLE time per EU is less than 1%

Avg. Chunk Read Hit Ratio Curves per TSU



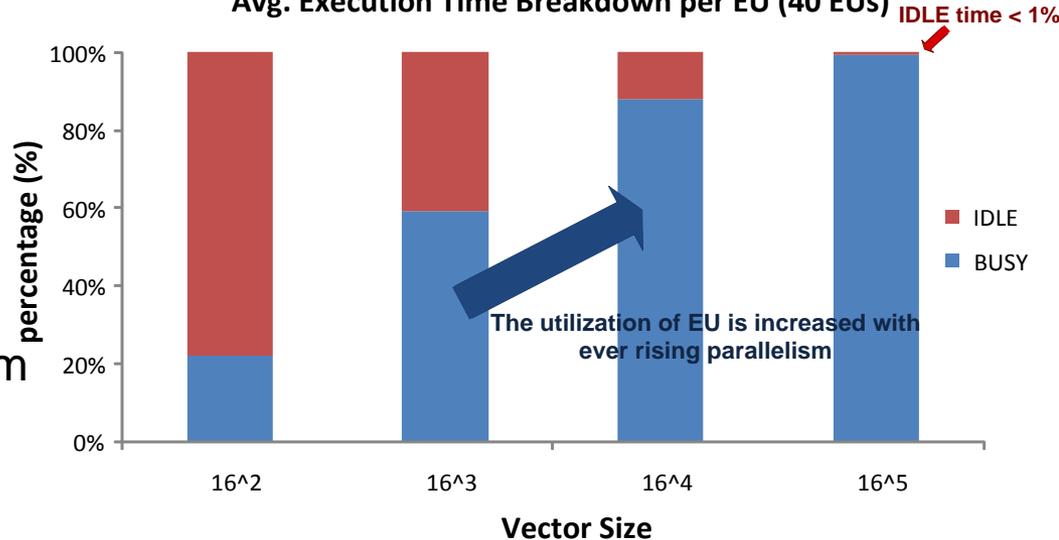
- ❖ Cache hit ratio of the top storage level is very low for the dot product program
- ❖ hit ratio is decreased with the increased vector size
- ❖ < 3% for vector size of 16^5

❖ Fortunately, ***a high hit ratio is not essential to achieve high utilization if there is lots of available parallelism***

❖ EU Utilization with Parallelism

❖ For problems with greater data locality, it can benefit from cache support for data reuse

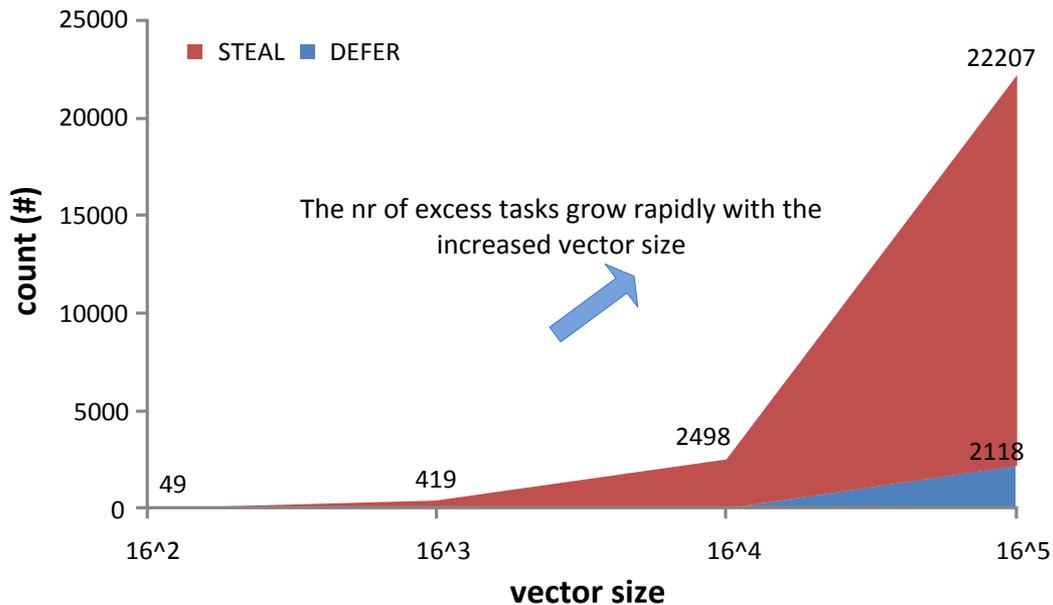
Avg. Execution Time Breakdown per EU (40 EUs)



The utilization of EU is increased with ever rising parallelism

IDLE time < 1%

The Growth of Excess Task (40 EUs)



- ❖ The *excess task* is defined as task that go through “*defer*” or “*steal*” process
- ❖ An *excess task* is available for stealing at runtime
- ❖ The number of excess tasks grows rapidly with increasing problem size

- ❖ How to manage excess tasks efficiently will become an important issue when Fresh Breeze principles are applied to a larger scale of massively parallel system
 - ❖ How to organize the task defer operation in a hierarchical, locality-aware way
 - ❖ How to integrate the factor of data locality into task stealing

Further Work

- Extrapolate results to hardware speeds.
- Test Matrix Multiply, which offers significant data reuse.
- Model a Three-Level Storage System including SSD (flash).
- Task distribution to improve data locality.
- FunJava Compiler: DFG transformations and code generation.
- Benchmark applications for testing and evaluation.