Willow: A User-Programmable SSD

Sudharsan Seshadri, Mark Gahagan, Sundaram Bhaskaran, Trevor Bunker, Arup De, Yanqin Jin, Yang Liu, and Steven Swanson

Non-Volatile Systems Laboratory

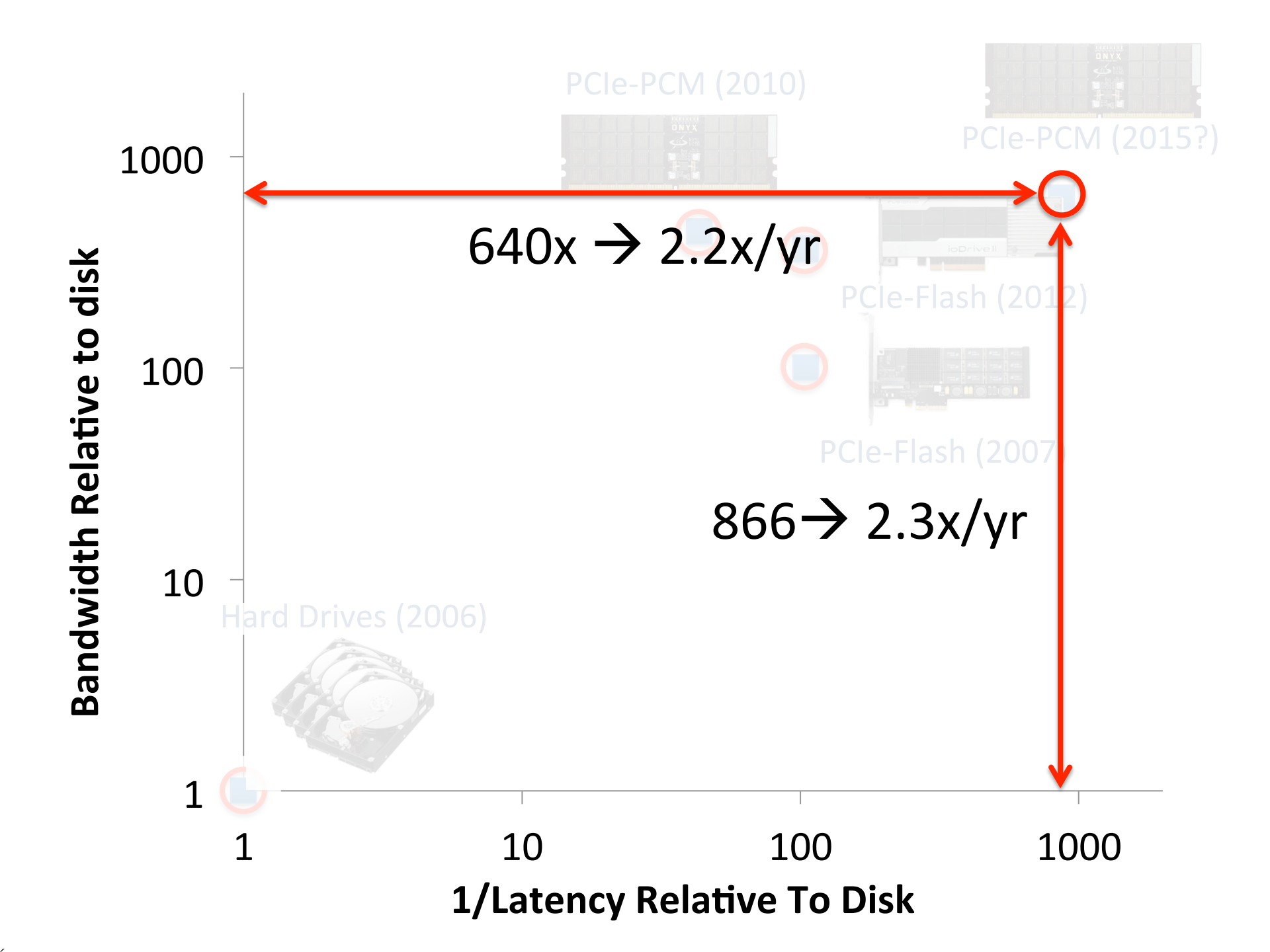
Computer Science and Engineering

University of California, San Diego

NVSL
Non-volatile Systems Laboratory

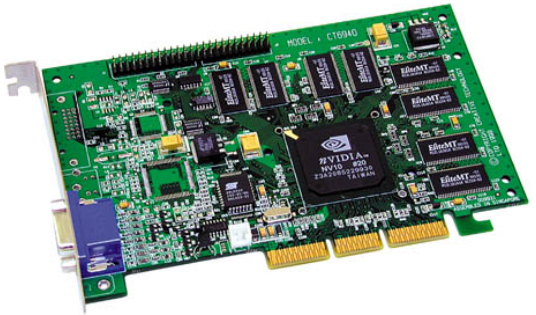
UCSDCSE
Computer Science and Engineering

UC San Diego

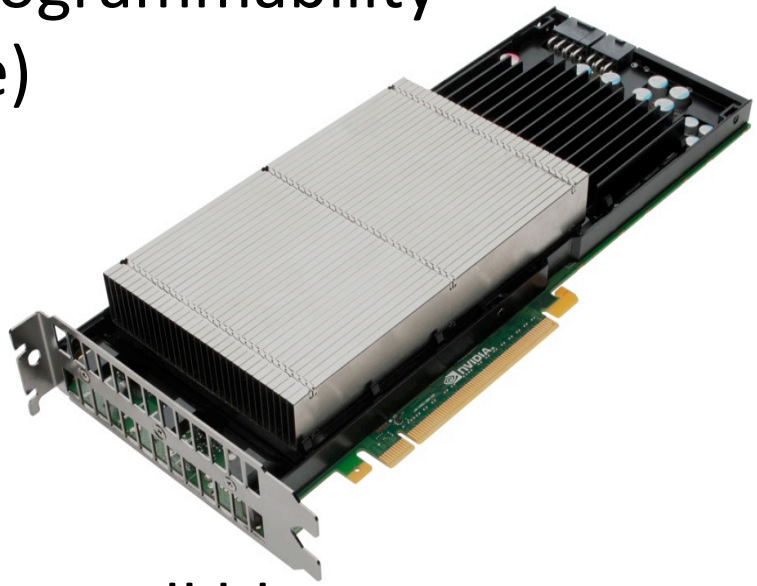

PCIe-PCM (2010)
PCIe-PCM (2015?)
640x → 2.2x/yr
PCIe-Flash (2012)
PCIe-Flash (2007)
866→ 2.3x/yr
Hard Drives (2006)
1000
100
10
1
Bandwidth Relative to disk
1
10
100
1000
1/Latency Relative To Disk

Case Study: Programmable GPUs

Hidden Programmability
(Firmware)



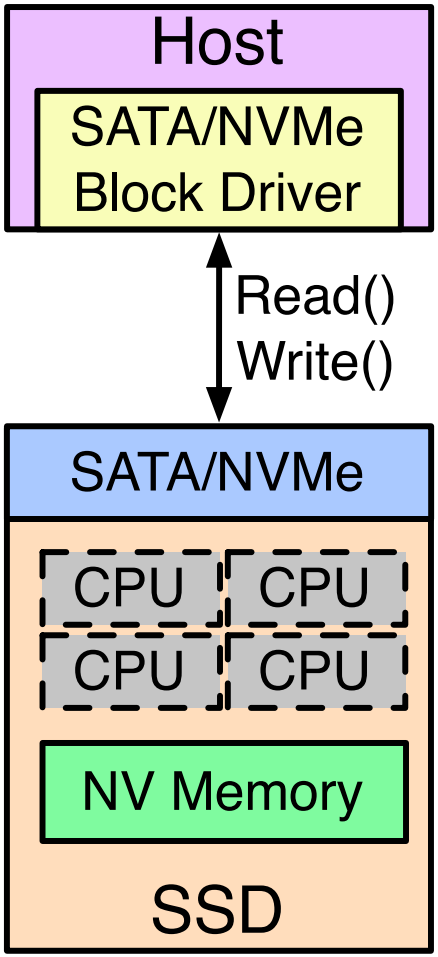
Partial Programmability
(Shaders)



Full blown Programmability

Modern SSDs Hide Their Programmability

- Fixed interface
  - SATA or NVMe
  - Storage-centric operations
- Flexible hardware
  - Multi-core processors
  - Complex firmware



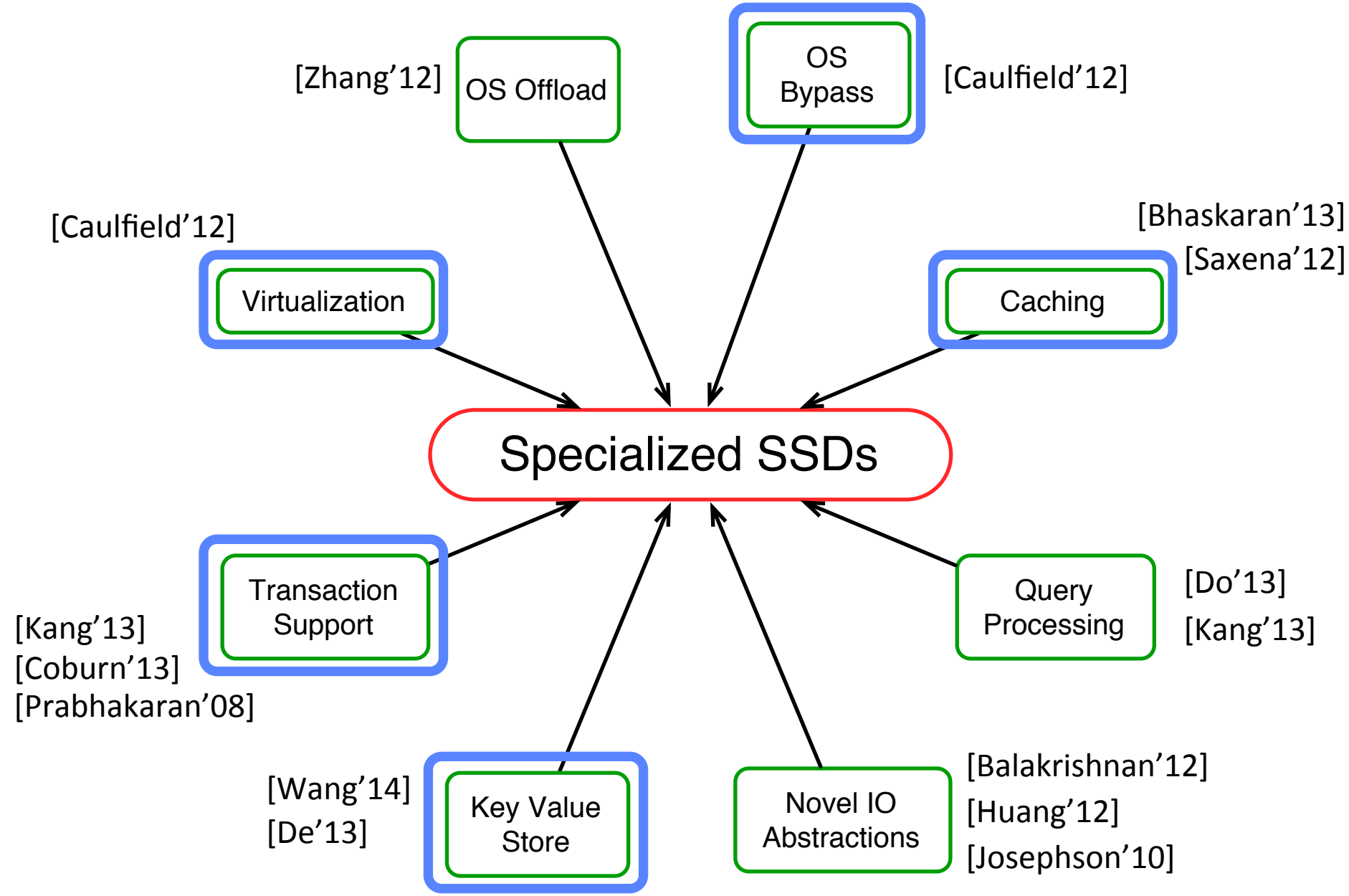
Candidates for Near-Storage Compute

- Data-intensive computation
  - Database scans
  - Transcoding
  - Analytics

Modern SSD processors are inadequate

- Data-dependent accesses
  - e.g. pointer chasing
- Semantic extension
  - e.g. transactions
- Privileged execution
  - e.g. OS offload

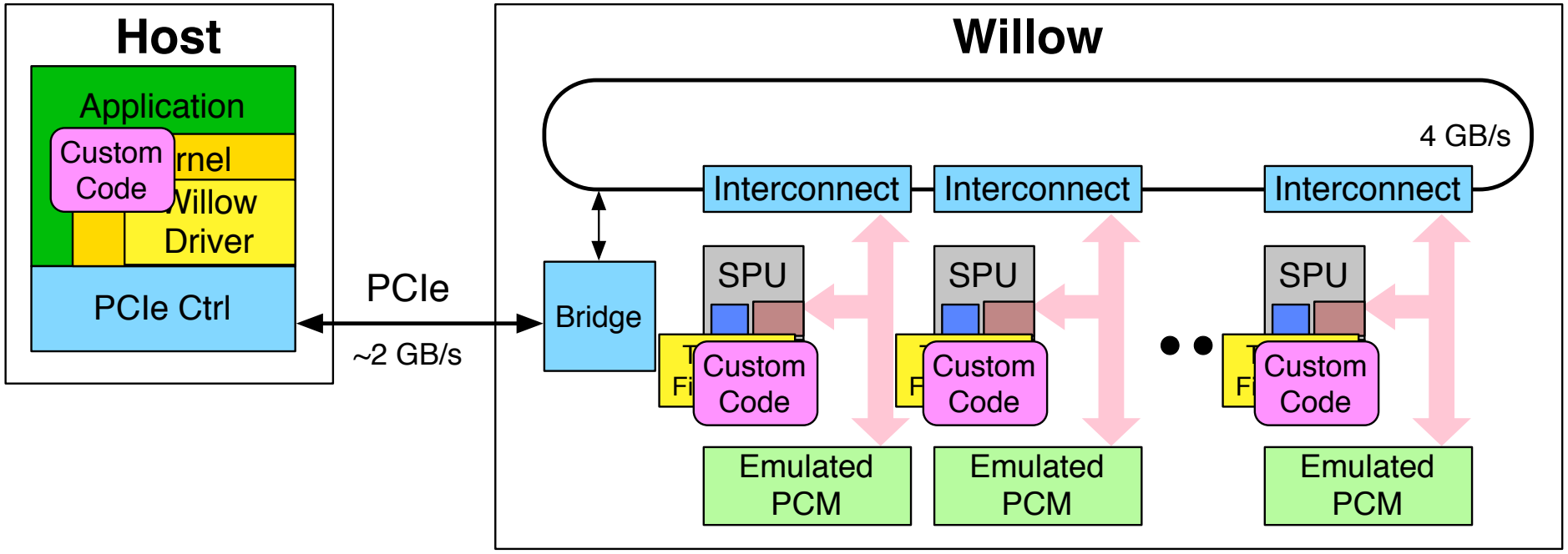
Feasible on modern SSD processors


[Zhang'12]
OS Offload
OS Bypass
[Caulfield'12]
[Caulfield'12]
Virtualization
[Bhaskaran'13]
[Saxena'12]
Caching
Specialized SSDs
Transaction Support
[Kang'13]
[Coburn'13]
[Prabhakaran'08]
Query Processing
[Do'13]
[Kang'13]
[Wang'14]
[De'13]
Key Value Store
Novel IO Abstractions
[Balakrishnan'12]
[Huang'12]
[Josephson'10]

A Programmable SSD Should…

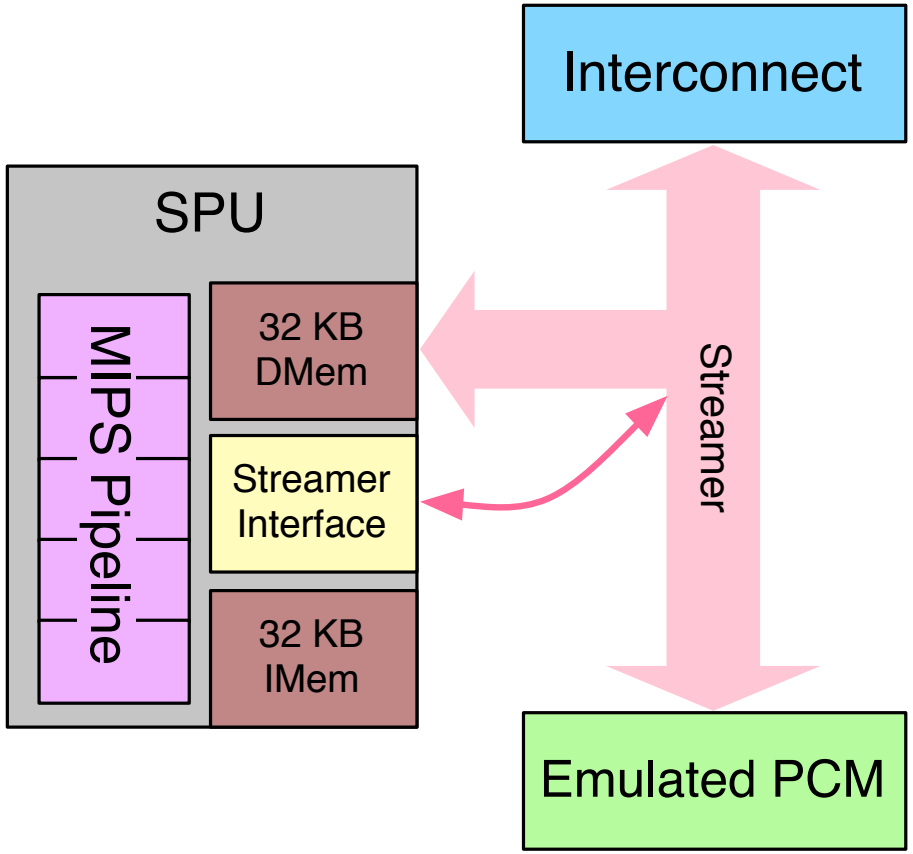
- Provide a flexible interface
  - New arguments, semantics, and operations
  - Programmable in C (or something better)
- Enforce file system permissions
- Allow execution of untrusted code
- Allow multiple specialized functions to coexist
- Allow for reuse and sharing of functions between applications
- Allow applications to invoke operations without a system call.
- Be able to run trusted code
  - The OS can delegate operations to Willow
  - Untrusted applications can to invoke them.

Willow System Overview



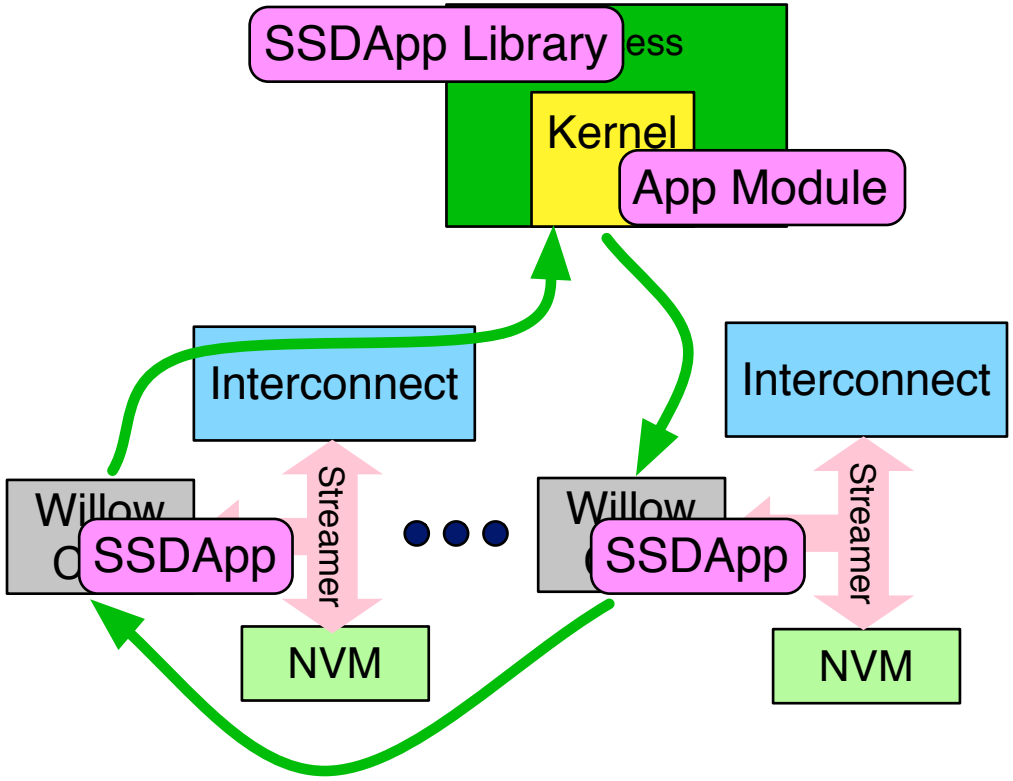
The Willow Processor Complex

- 125 MHz MIPS processor
- 32 KB of D- and I-mem
- A bank of NVM
- Network interface
- High-bandwidth Data Streamer



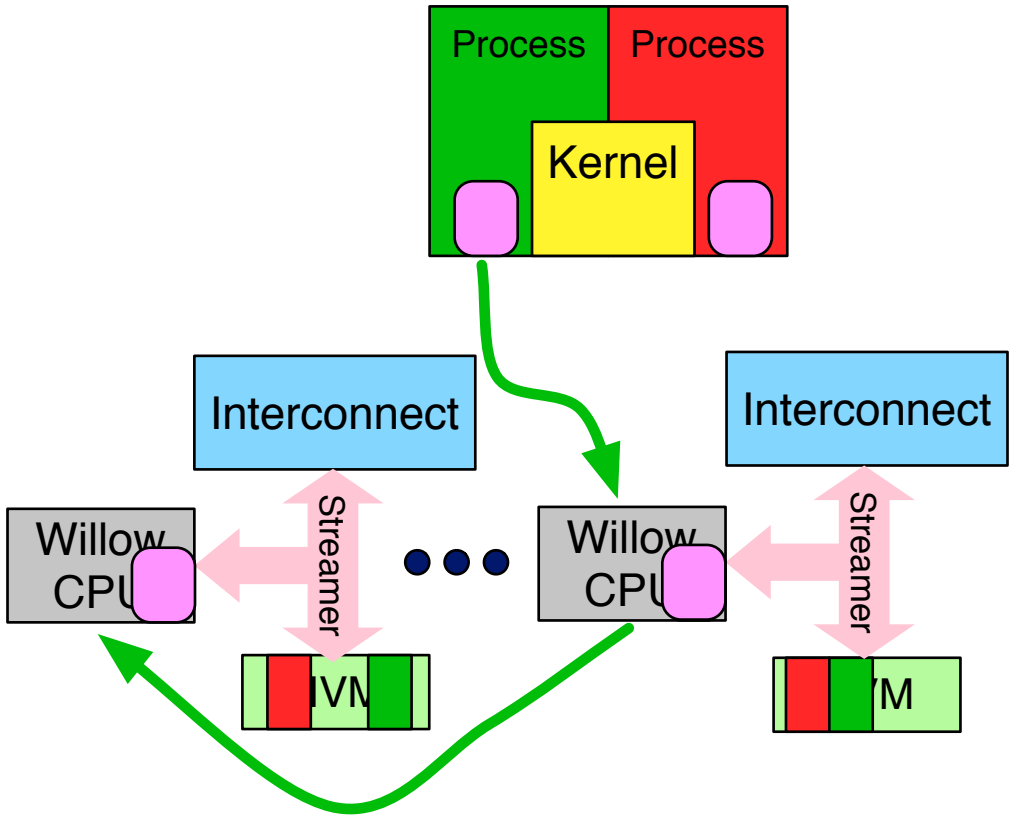
Willow Usage Model and SSD Apps

- The programmer creates an "SSD App"
- The kernel installs "SSDApps" for applications
  - The Willow-resident code
  - A userspace library
  - A kernel module, if needed
- Communication via RPCs
- Host and SSD code can send and receive RPCs



Trust and Protection

- A file system sets protection policy
- RPCs carry an unforgeable ProcessID
- Execution at SPUs is always on behalf of a ProcessID
- The Willow driver installs access rights
- Willow firmware checks permissions on access



Willow Case Studies

- Basic IO
- Direct IO

Standard Equipment

- Caching
- Transaction processing
- Key-Value Store
- File Append w/o the file system

Transaction Acceleration with MARS

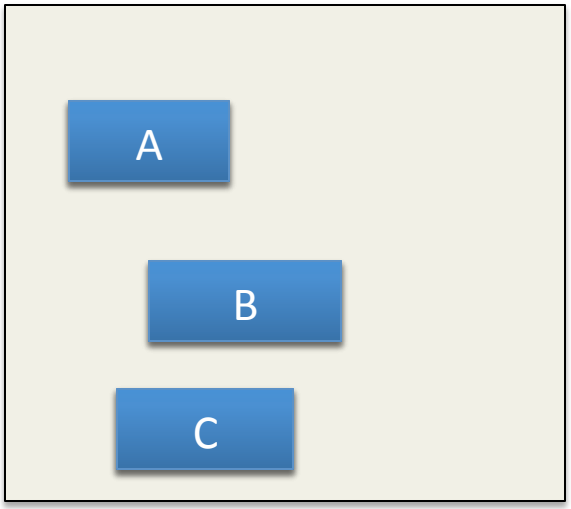
[SOSP'13]

UC San Diego

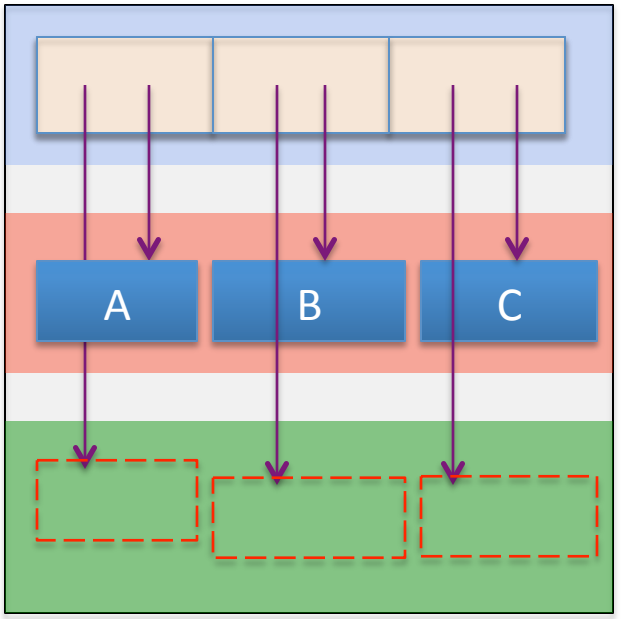
Editable Atomic Writes in Willow

```
LogWrite(bufA,addrA,lenA,logAddrA);
LogWrite(bufB,addrB,lenB,logAddrB);
LogWrite(bufC,addrC,lenC,logAddrC);
Commit();
```

Host Memory



Willow

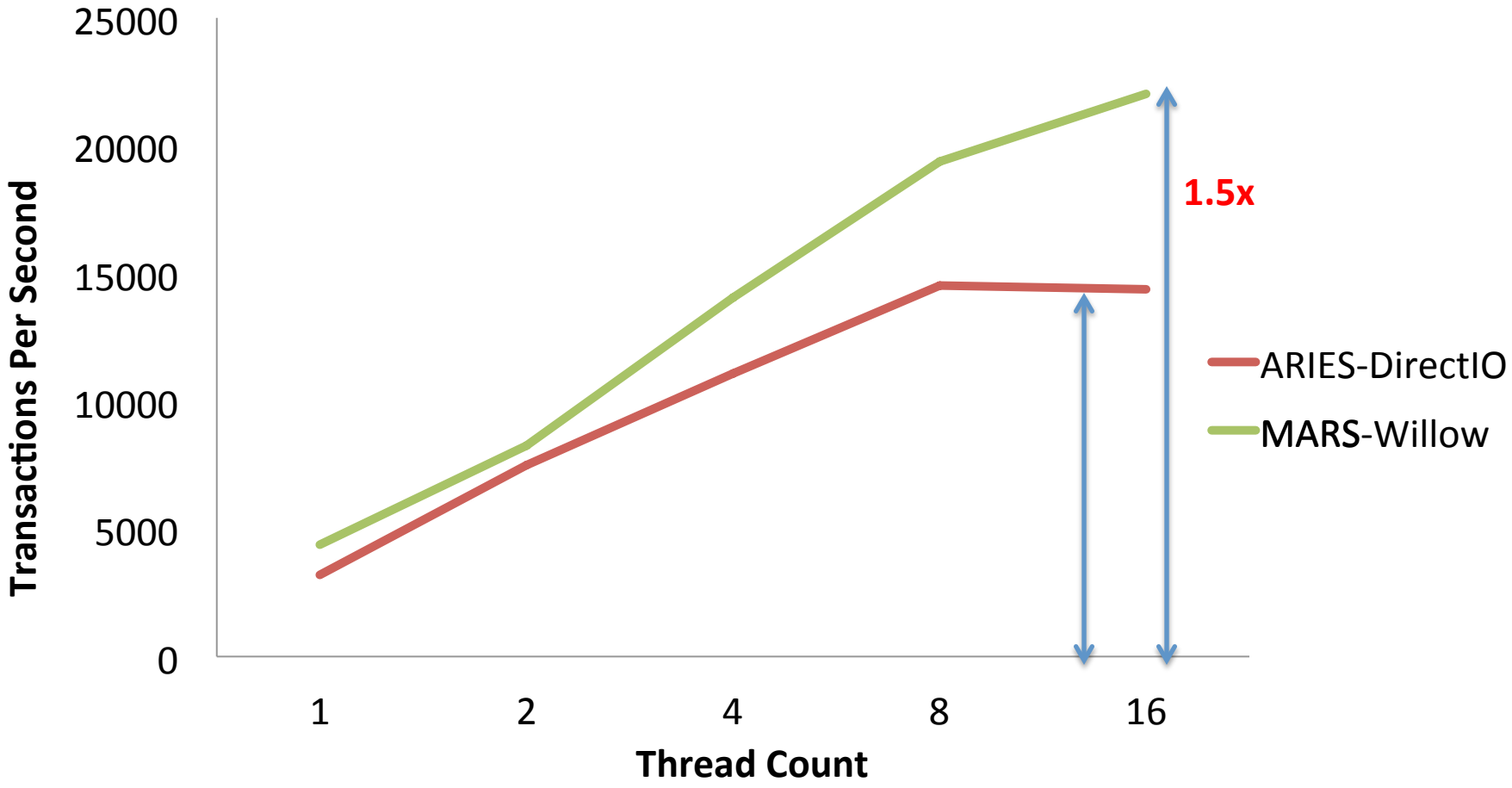


Metadata File

Log File

Data File

Performance Benefits On TPCB



Observations and Limitations

- SSD App development is relatively easy
- Composability of SSD Apps is very valuable
- Striping data across SPUs increases complexity for some SSD Apps
- Limited instruction and data storage at SPUs is a persistent challenge

The time is ripe for programmable storage

- Fast NVMs increase storage flexibility and performance demands
- Existing SSDs are already "software defined"
- Numerous applications already exist
- Willow provides a clean, flexible interface
  - Smooth integration with existing software
  - Powerful enough for complex applications
  - Preserves file system protections
- Programmable storage can simplify and accelerate applications

Thanks!

cns
center for networked systems

NVSL
Non-volatile Systems Laboratory

UC San Diego

CFAR

UCSDCSE
Computer Science and Engineering