



# Merge Framework

A Programming Model for  
Heterogeneous Multi-core Systems

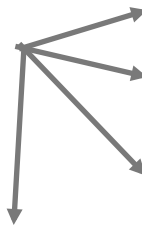
Michael Linderman

[mlinderm@stanford.edu](mailto:mlinderm@stanford.edu)

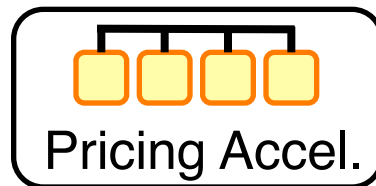
Stanford University, Stanford, CA

# Motivating Example: FastFinance Inc.

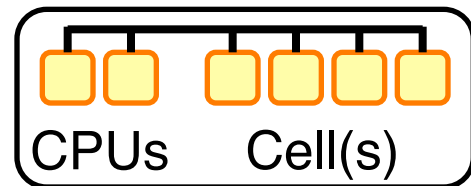
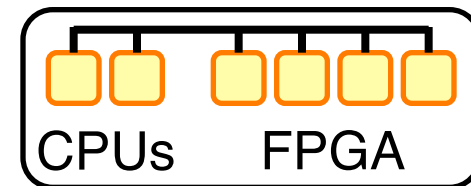
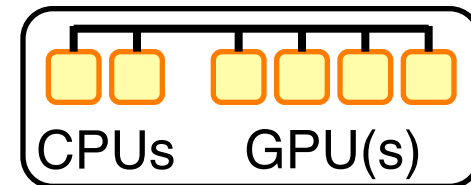
Your company's computation-intensive killer app



A HW startup's ultra-fast derivatives pricing chip

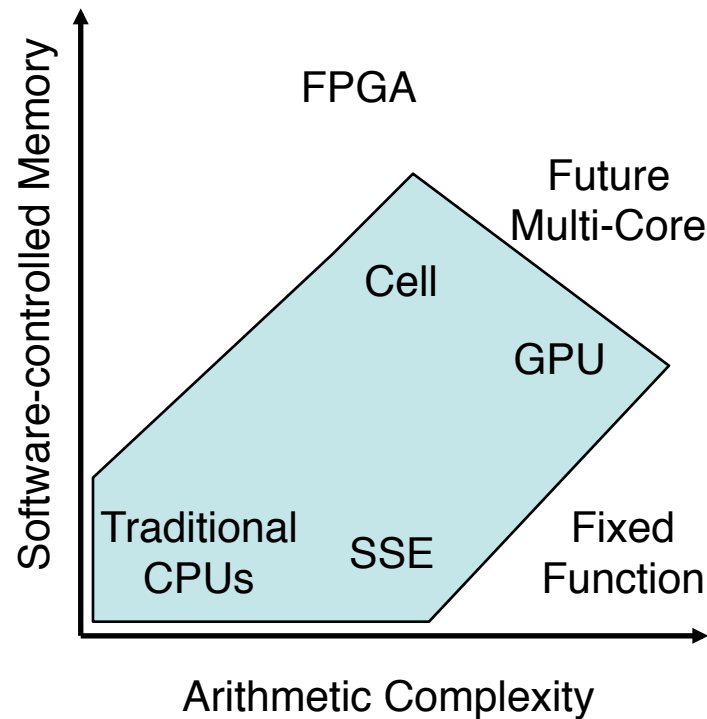


## Customer Platforms



**How do write your application to achieve the best performance across many different systems?**

# Existing Approaches to Heterogeneity

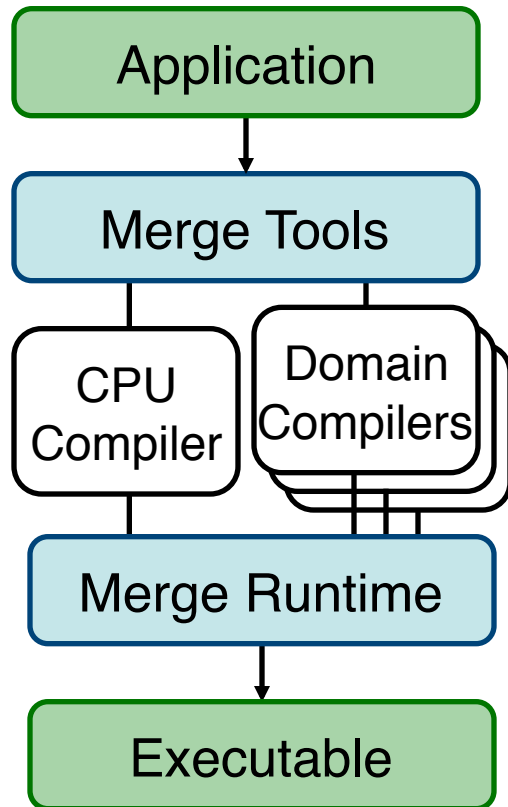


**Restrict the design space**

```
#ifdef GPU
  if (length > 500)
    doGPU()
  else
    doCPU()
#else
  . . .
```

**Ad hoc mix of static and dynamic code selection**

# Merge Approach



No one compiler is the exclusive source of the executable  
Provide single mechanism for static and dynamic code selection

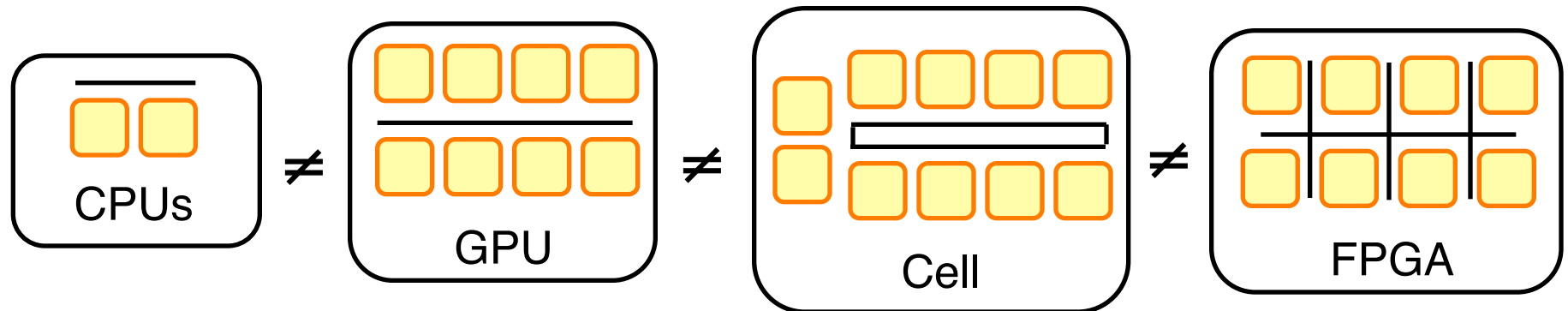
# Merge in a Nutshell

- Applications are architecture-agnostic
- Merge automatically maps between app and arch-specific implementation
  - Separate solving the problem from selecting correct/best implementations
  - Provide minimal selection system to ensure correctness
  - Expose APIs to build more complex/better mappings
- Ideal: Merge apps equal to custom apps on any platform

# How to Enable Arch. Agnosticism

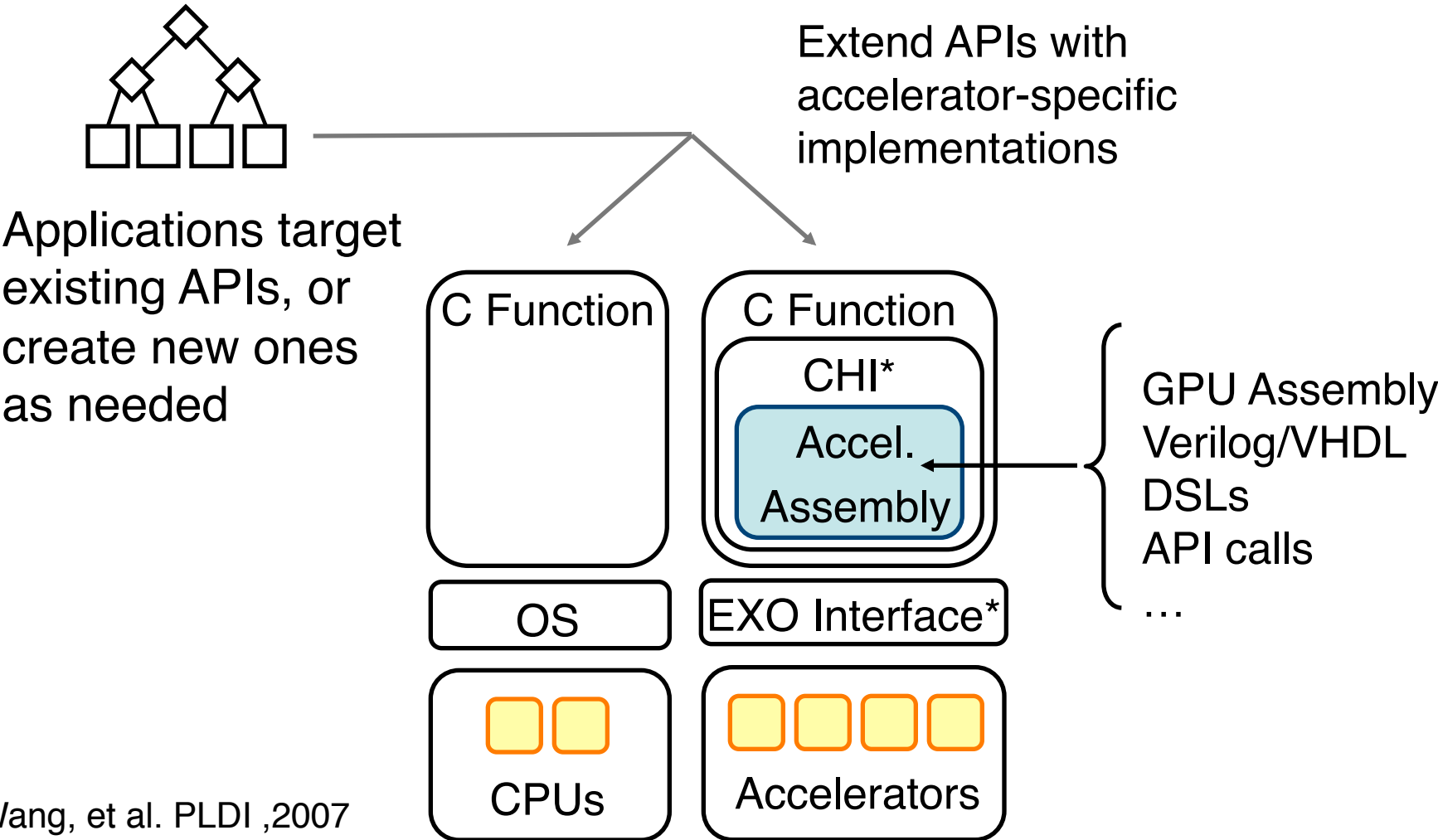
- Inclusive, extensible, compiler primitives
- Uniform interface across accelerators
- Indirection between app and implementation
- Easy addition of new/different implementations
- Invariants flow from function implementations to apps

# Primitives: Don't Guess, Don't Compromise



- Targeted arch are all very different
- No fixed set of primitives can abstract them all
- Languages already have extensible primitives - *functions*

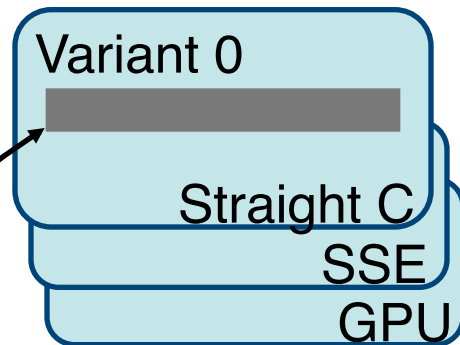
# Uniform Accelerator Abstraction



\*Wang, et al. PLDI ,2007

# Generic Functions

Target & Granularity Variants



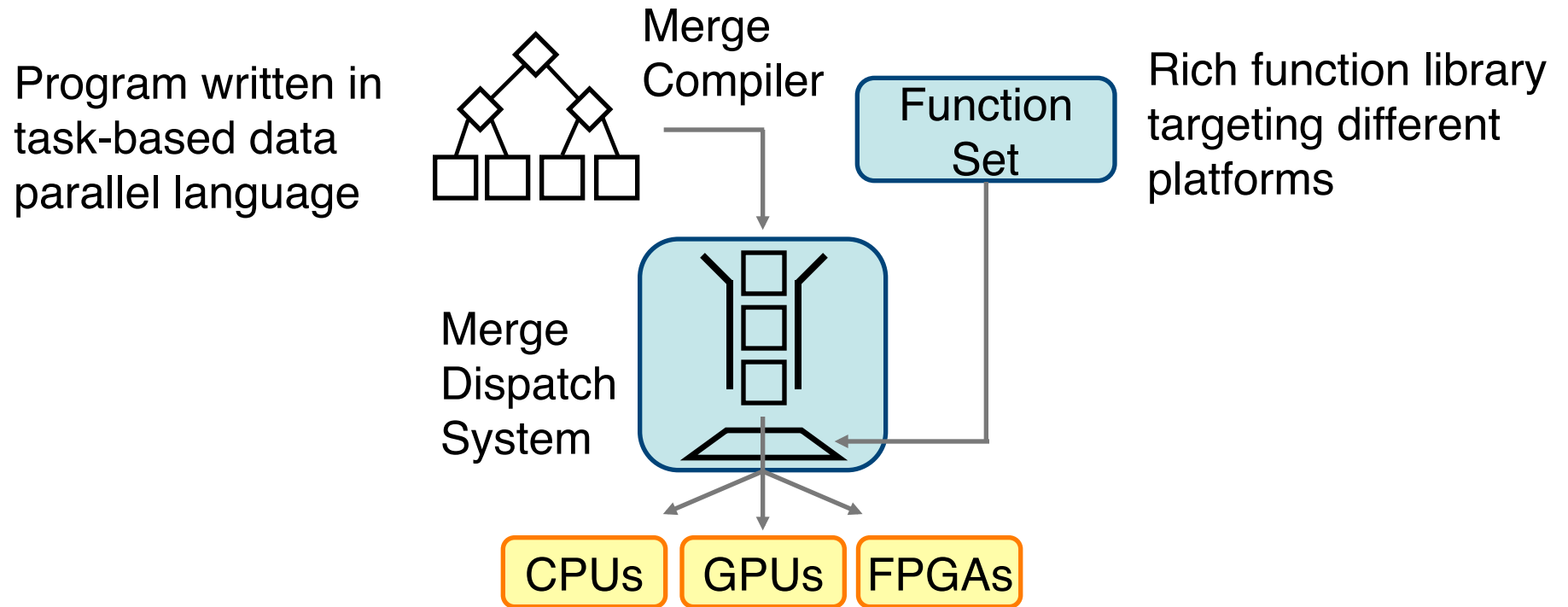
```
predicate(in0.cols == 8);  
predicate(in0.rows > 512);
```

Generic Function

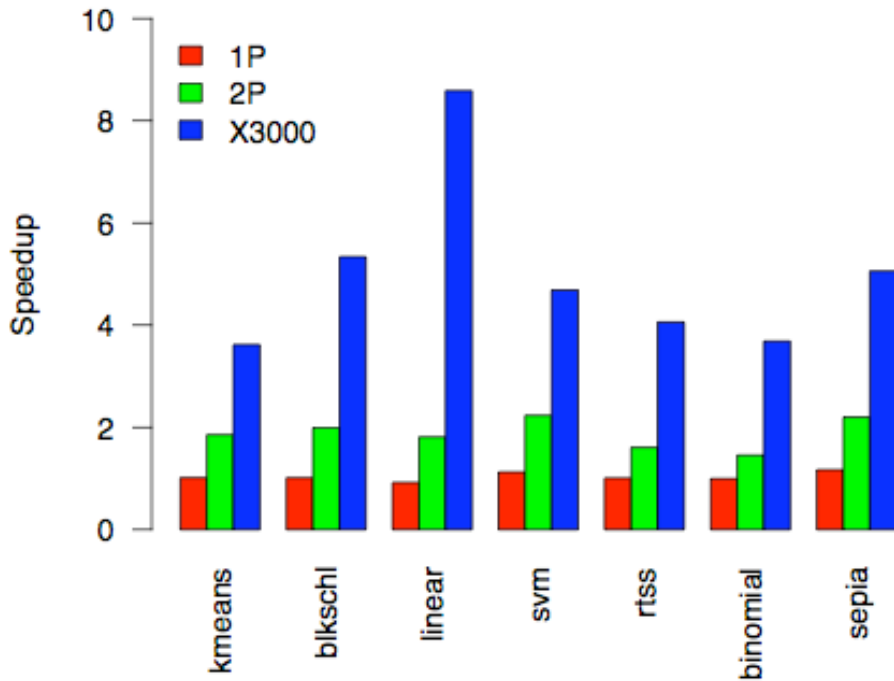
```
if (in0.cols == 8 &  
    in0.rows > 512) {  
    variant0(...);  
}  
else if (...) {}  
    . . .  
else {}
```

Programmer-supplied predicates annotate functions  
Function variants are ‘bundled’ to create generic functions

# Merge Parallel Programming



# Performance



**Speedup vs. reference C code for 1 and 2 CPU cores and 2 CPU cores + GPU**

## Heterogeneous

*2 Cores: 1.4x - 2.4x*

*2 Cores + GPU: 3.6x - 8.5x*

## Homogeneous

*32-way SMP: 5.2x - 22x*

## Cooperative multi-threading

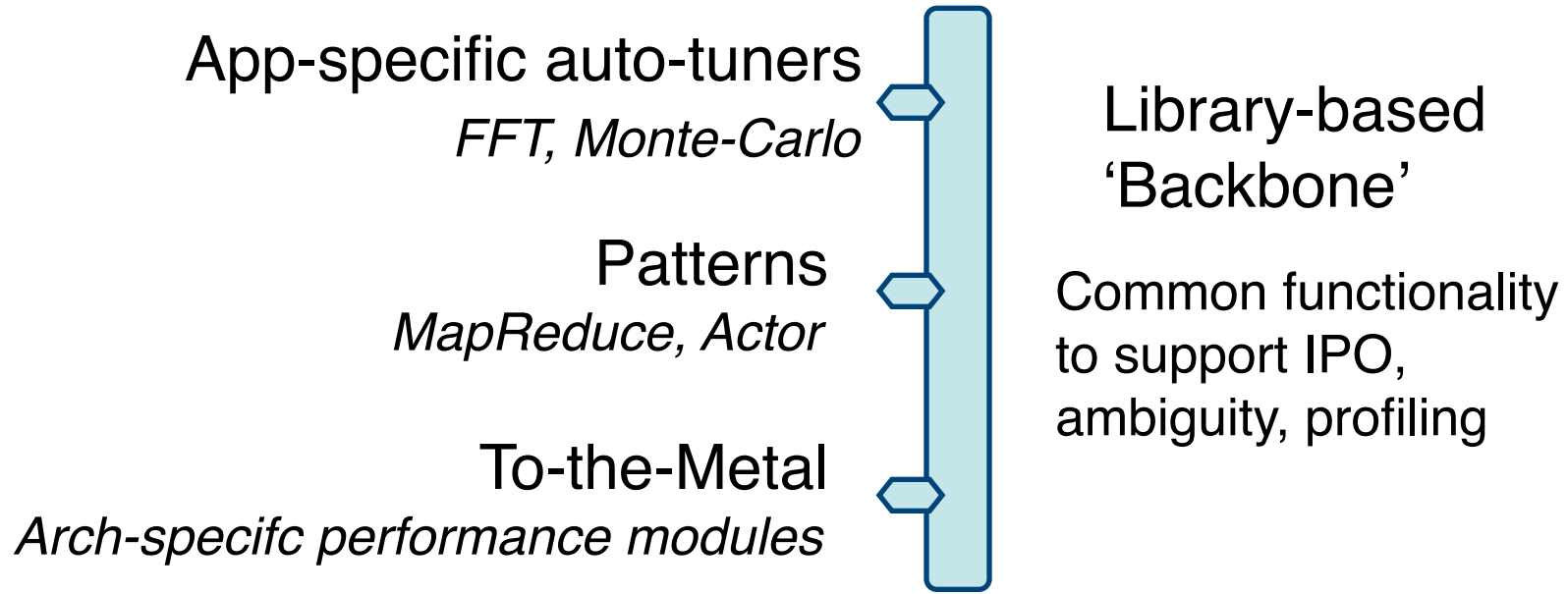
Addition of CPU threads to GPU-only execution adds 25-50% speedup per core

# A Few Lessons ...

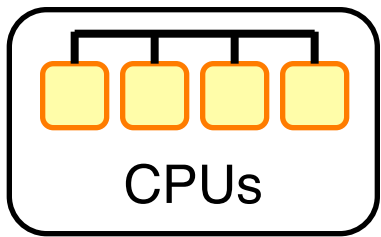
- Democratize access to the executable  
Code for HW in whatever way is appropriate
- Introspection of generic functions is powerful  
Tools at different abstraction levels can use the same information to different ends
- Granularity strongly influences performance  
Need additional tool support to synthesize coarser-grain implementations
- Locally controlled scheduling can be tricky  
Couldn't completely eliminate programmer-supplied global scheduling information

# Merge Framework

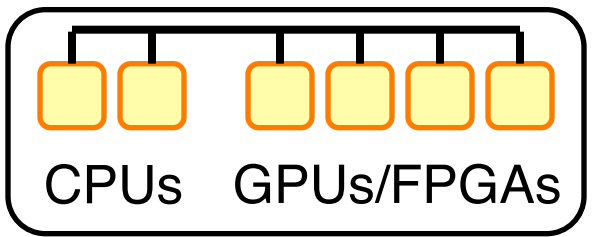
Informatics Applications  
Bio, Finance, Entertainment, Security



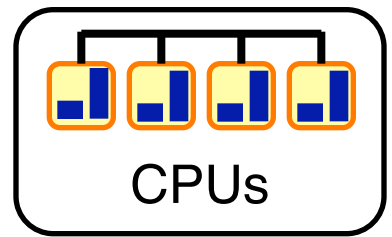
OS / Hypervisor



Homogeneous Multi-core



Heterogeneous Multi-core



Multi-core with Heterogeneous uArch

# Thanks

- Advisor: Prof. Teresa Meng
- Funding: FCRP (C2S2)
- At Intel
  - Hong Wang, Jamison Collins, Perry Wang, Ethan Schuchman, Anne Bracy, Ghassan Yacoub
- At Stanford
  - James Balfour, Omid Azizi
- For more information:
  - M. D. Linderman, J. D. Collins, H. Wang, T. H. Meng, “Merge: A Programming Model for Heterogeneous Multi-Core Systems”, Proc. of ASPLOS, pp. 287-296, 2008.
  - Available at <http://www.stanford.edu/~mlinderm> or e-mail me at [mlinderm@stanford.edu](mailto:mlinderm@stanford.edu)