codeplay®

THE HETEROGENEOUS SYSTEMS EXPERTS

Performance Portability & PRODUCTIVITY

Gordon Brown & Michael Wong

CppCon 2020 – Sep 2020

- Learning objectives:
    - Learn about the Iron triangle of parallel computing
    - Learn about the goals of parallel computing
    - Lean about how to iterate through parallel programming tasks

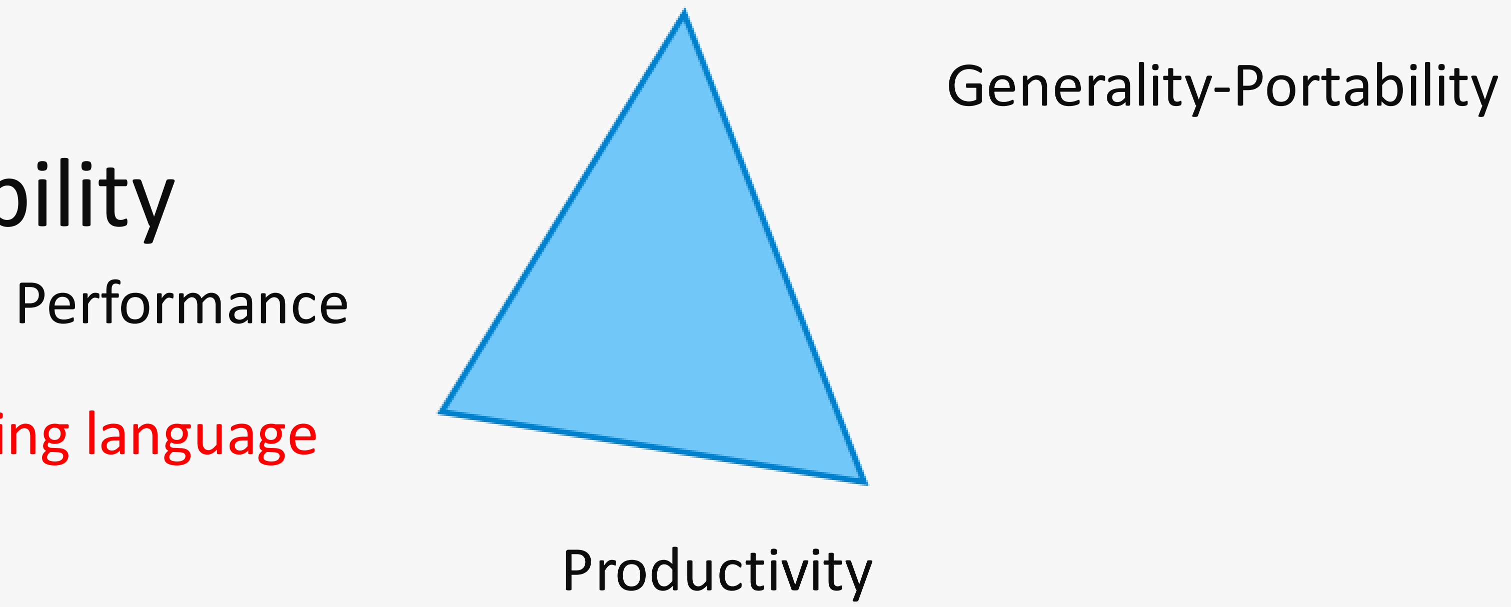
Step Back

So what are the goals of multithreaded parallel concurrent heterogeneous programming

And how do you do it in general?

So What are the Goals?

- Goals of Parallel Programming over and above sequential programming
    1. Performance
    2. Productivity
    3. Generality-Portability



The Iron Triangle of Parallel Programming language nirvana

- Oh, really??? What about correctness, maintainability, robustness, and so on?

- And if correctness, maintainability, and robustness don’t make the list, why do productivity and generality?

- Given that parallel programs are much harder to prove correct than are sequential programs, again, shouldn’t correctness really be on the list?

- What about just having fun?

Performance

- Broadly includes scalability and efficiency
- If not for performance why not just write sequential program?
- parallel programming is primarily a performance optimization, and, as such, it is one potential optimization of many.

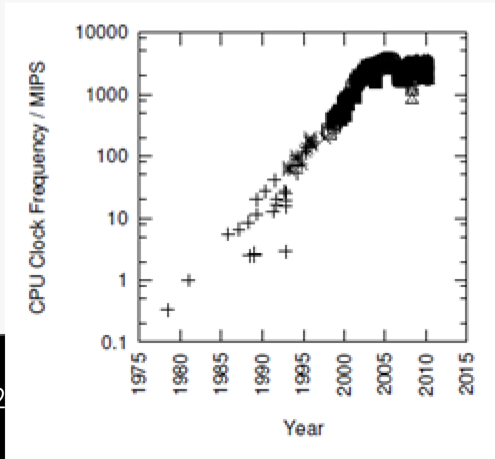
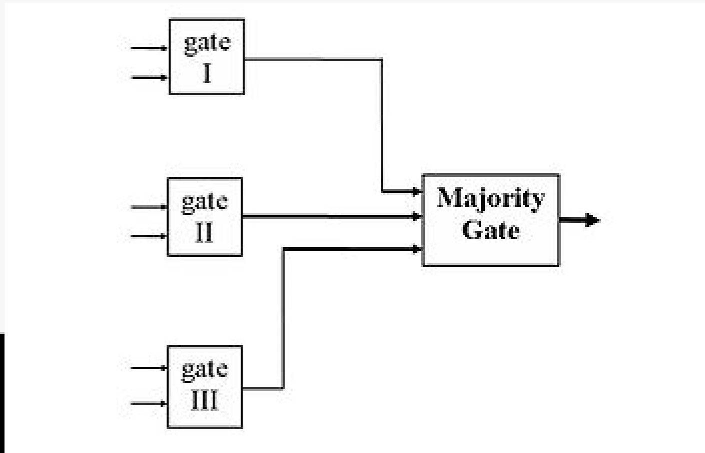


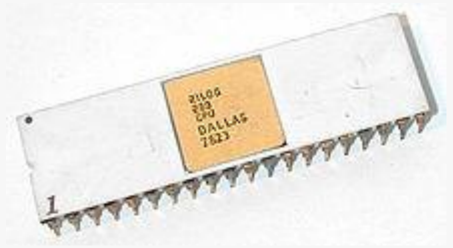
Diagram thanks to Paul McKenney

- Are there no cases where parallel programming is about something other than performance?



Productivity

- Perhaps at one time, the sole purpose of parallel software was performance. Now, however, productivity is gaining the spotlight.



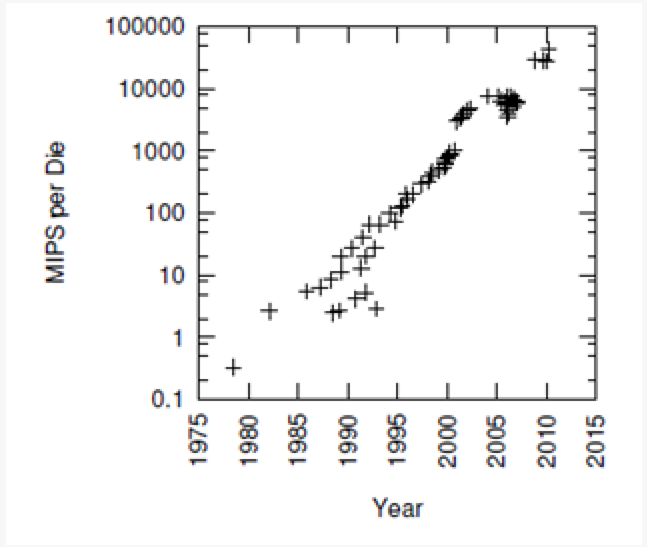
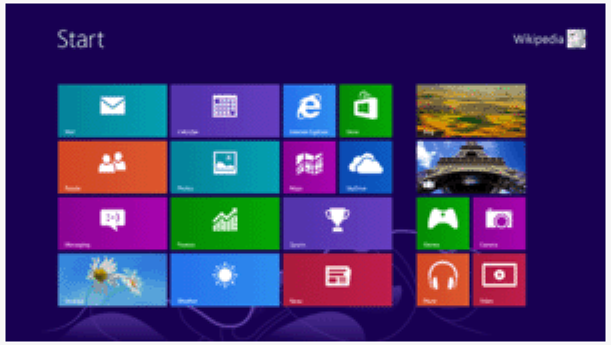


Diagram thanks to Paul McKenney

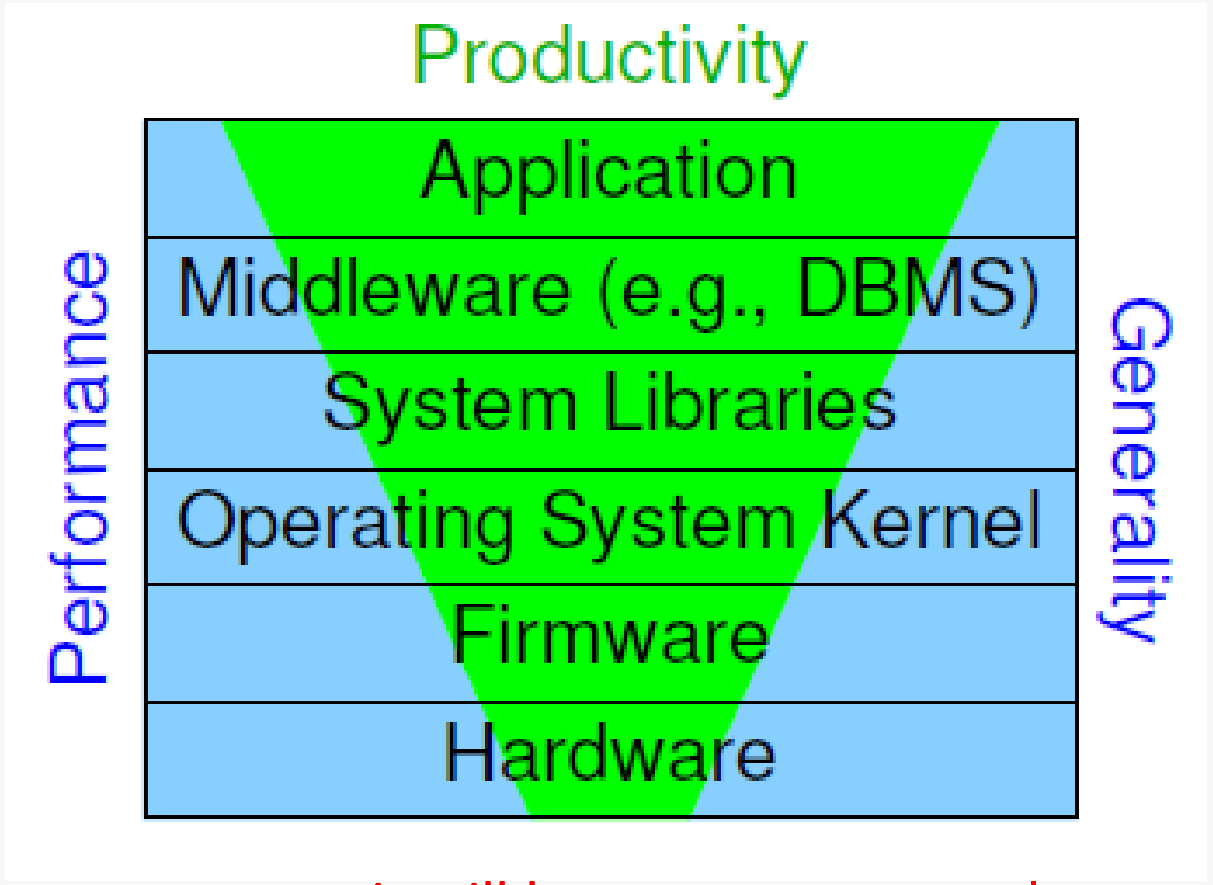
- Why all this prattling on about non-technical issues???And not just any non-technical issue, but productivity of all things? Who cares?



- Given how cheap parallel systems have become, how can anyone afford to pay people to program them?

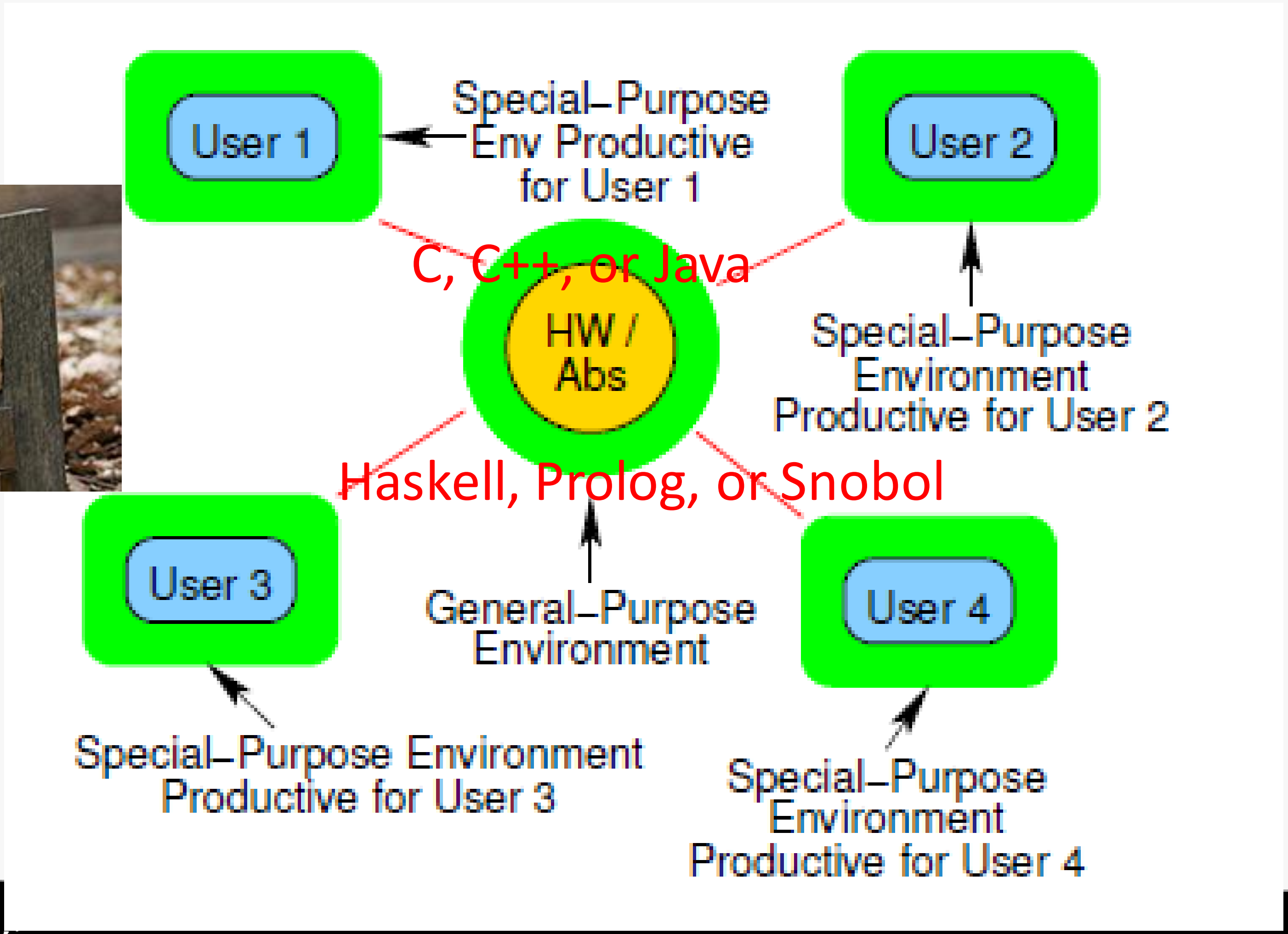
Generality (Portability)

- C/C++ locking+threads
- Java
- MPI
- OpenMP
- SQL



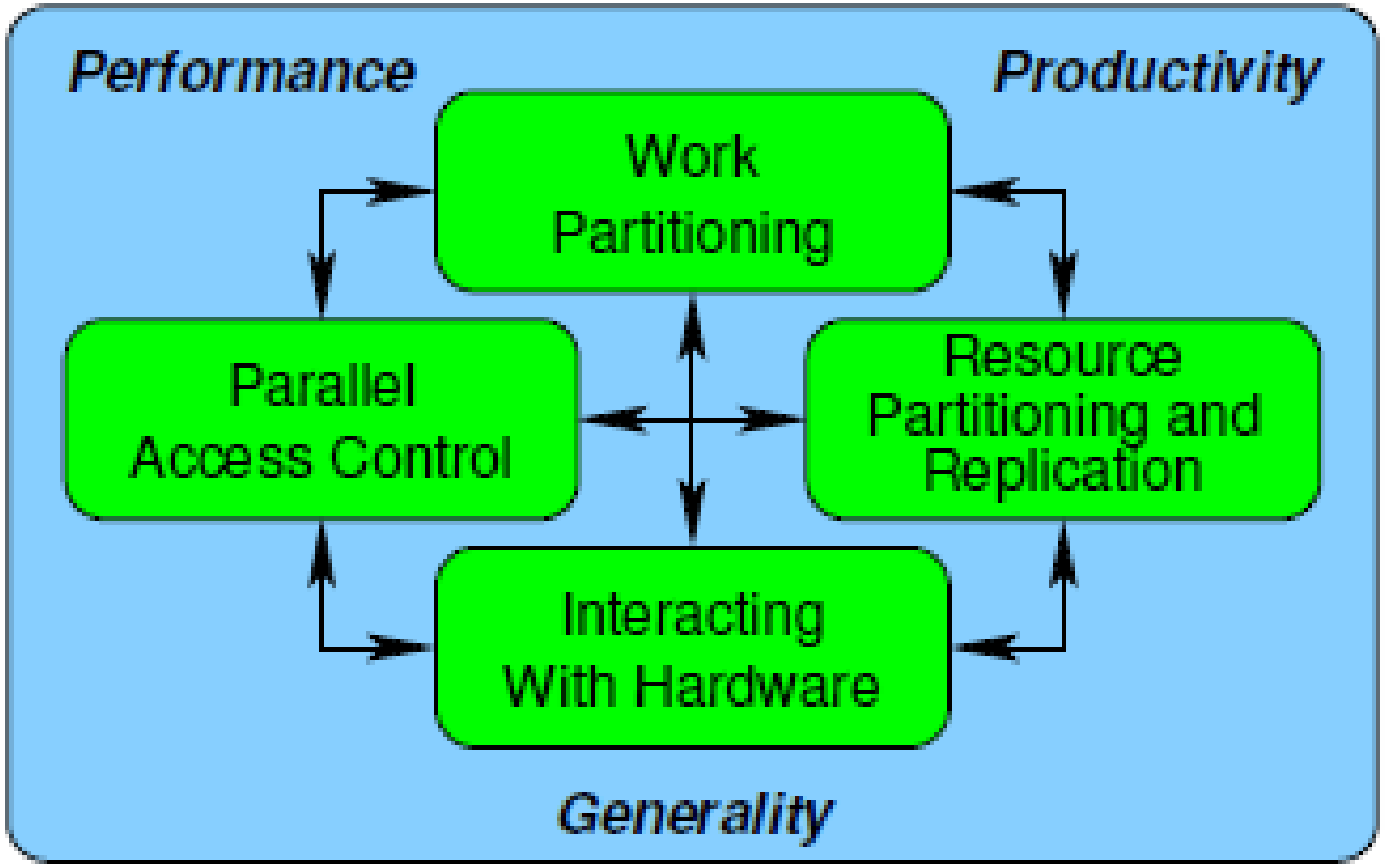
Until such a nirvana appears, it will be necessary to make engineering tradeoffs

Tradeoff Generality and Performance


User 1
Special–Purpose Env Productive for User 1
User 2
C, C++, or Java
HW / Abs
Special–Purpose Environment Productive for User 2
Haskell, Prolog, or Snobol
User 3
General–Purpose Environment
User 4
Special–Purpose Environment Productive for User 3
Special–Purpose Environment Productive for User 4

- This is a ridiculously unachievable ideal! Why not focus on something that is achievable in practice?

Parallel programming tasks



Work Partitioning

- Greatly increase performance, scalability but can greatly increase complexity
- permitting threads to execute concurrently greatly increases the program's state space, which can make the program difficult to understand and debug
  - Can decrease productivity

- Other than CPU cache capacity, what might require limiting the number of concurrent threads?

Parallel Access Control

- Does access depend on resource location?
- How does thread coordinate access to the resource?

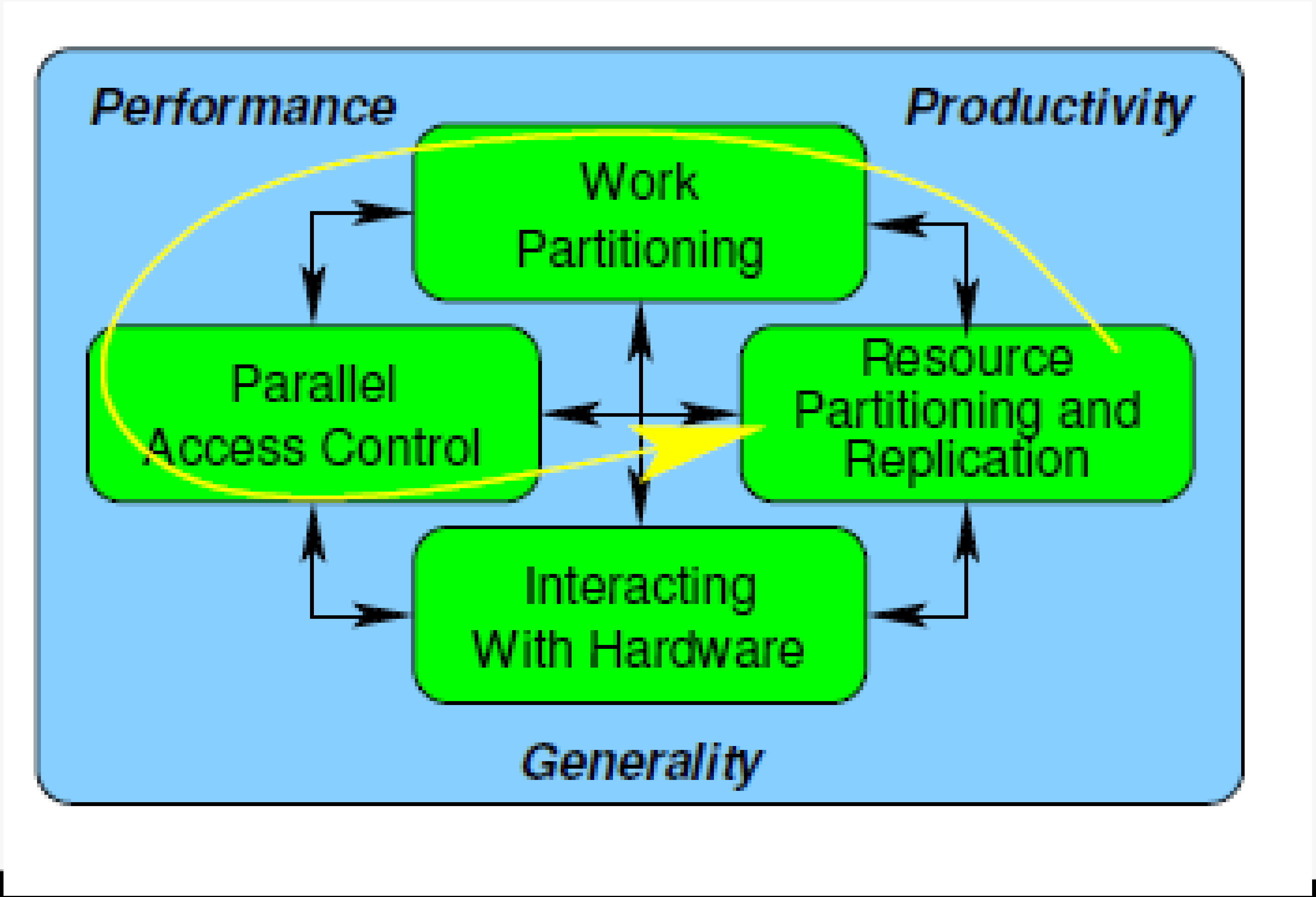
Resource partitioning and replication

- Data may be partitioned over computers, disks, NUMA nodes, CPU cores, pages, cache lines, instances of synchronization primitives, or critical section of code
- Resource partitioning is frequently application dependent

Interacting with Hardware

- developers working with novel hardware features and components will often need to work directly with such hardware
- direct access to the hardware can be required when squeezing the last drop of performance out of a given system

Composite Capabilities



Key takeaways

Iron Triangle of Parallel programming language

Remember how to iterate through parallel programming tasks

codeplay®
THE HETEROGENEOUS SYSTEMS EXPERTS

Questions?