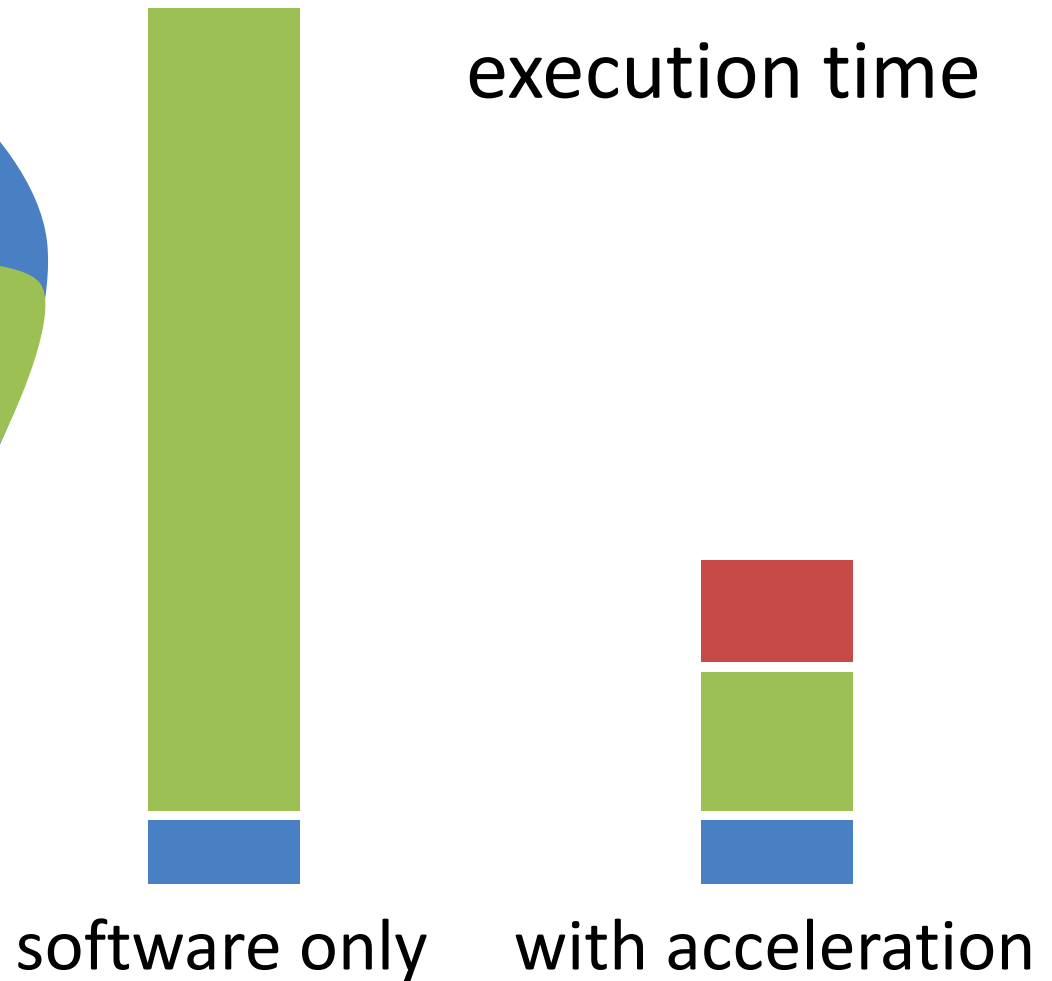
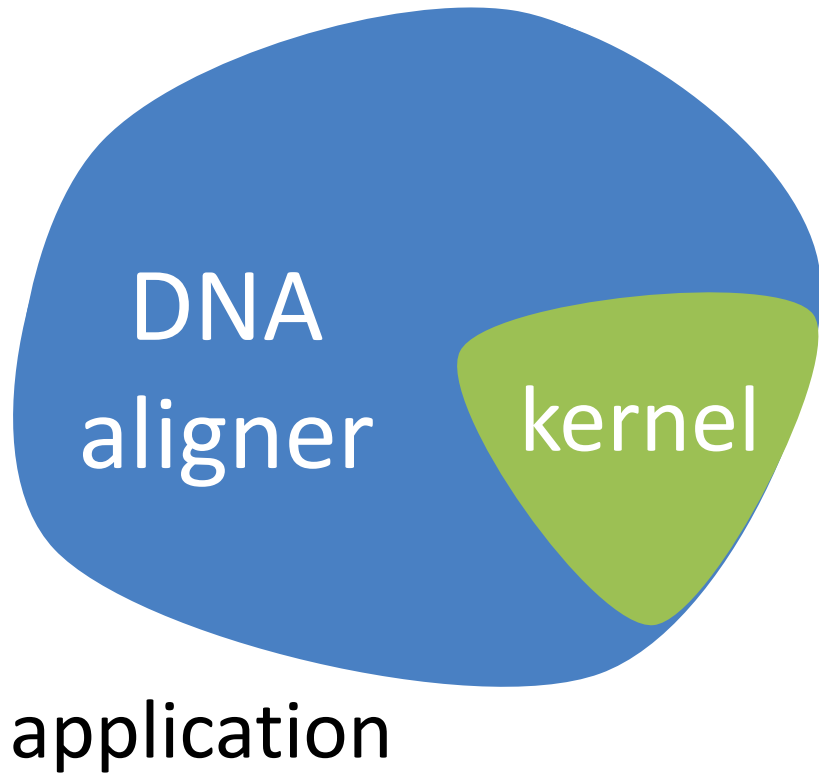


Dynamic Memory Allocation Strategies for Hybrid Architectures

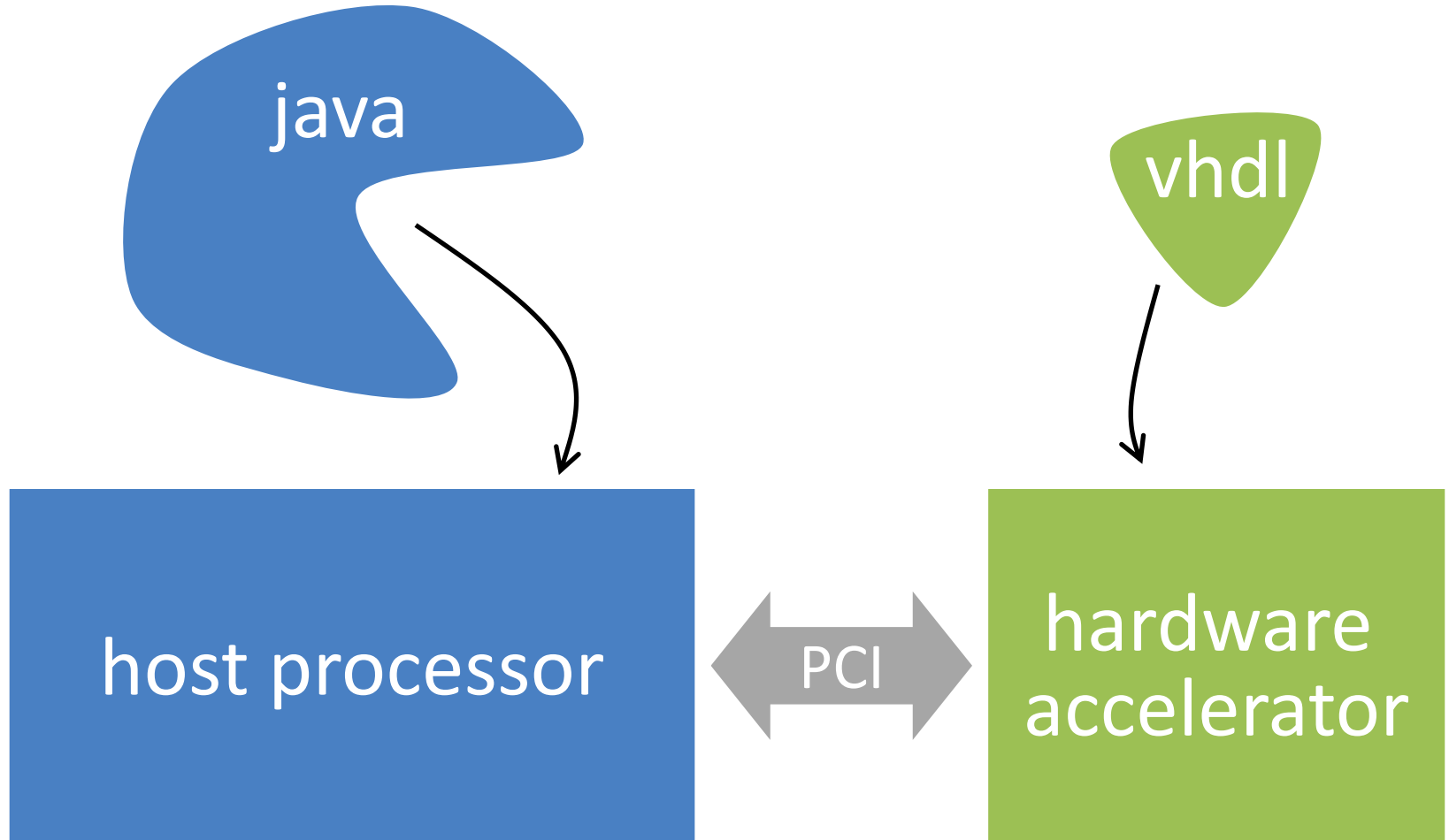
Peter Bertels

Wim Heirman and Dirk Stroobandt

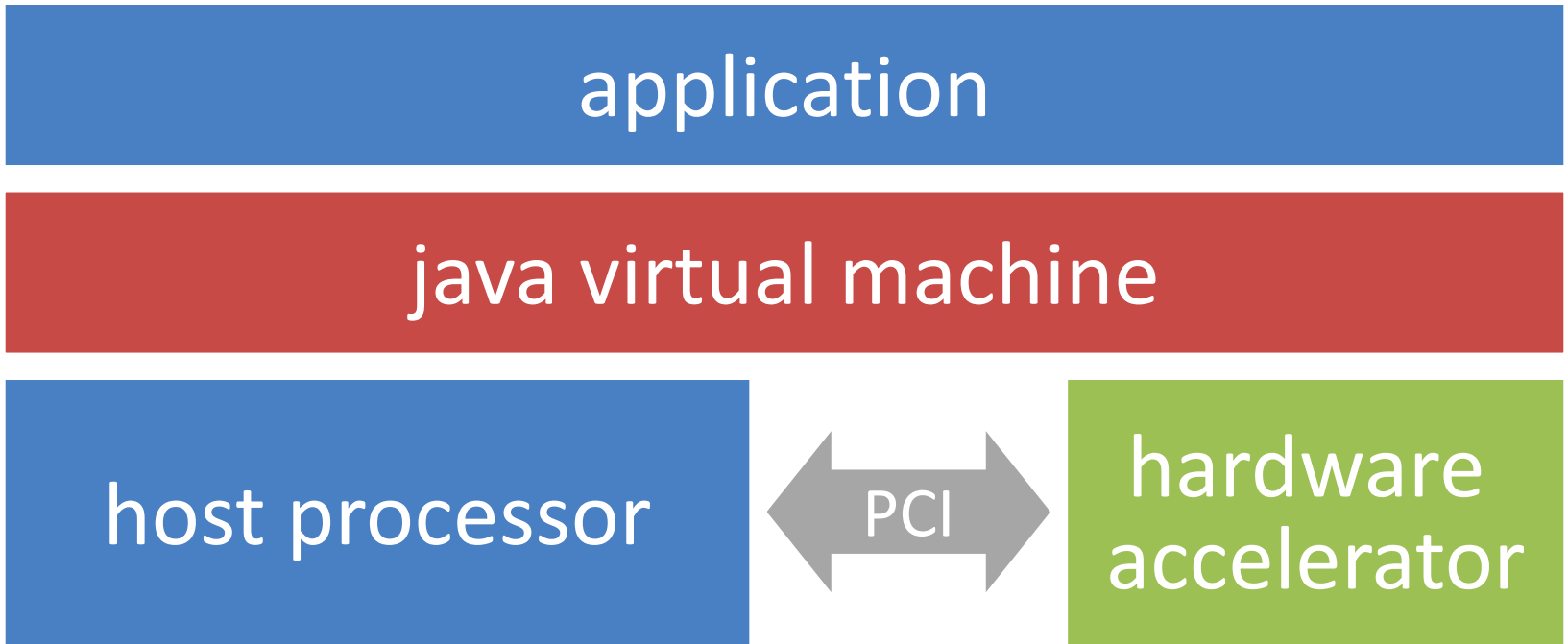
Hardware accelerators may lead to a significant performance boost



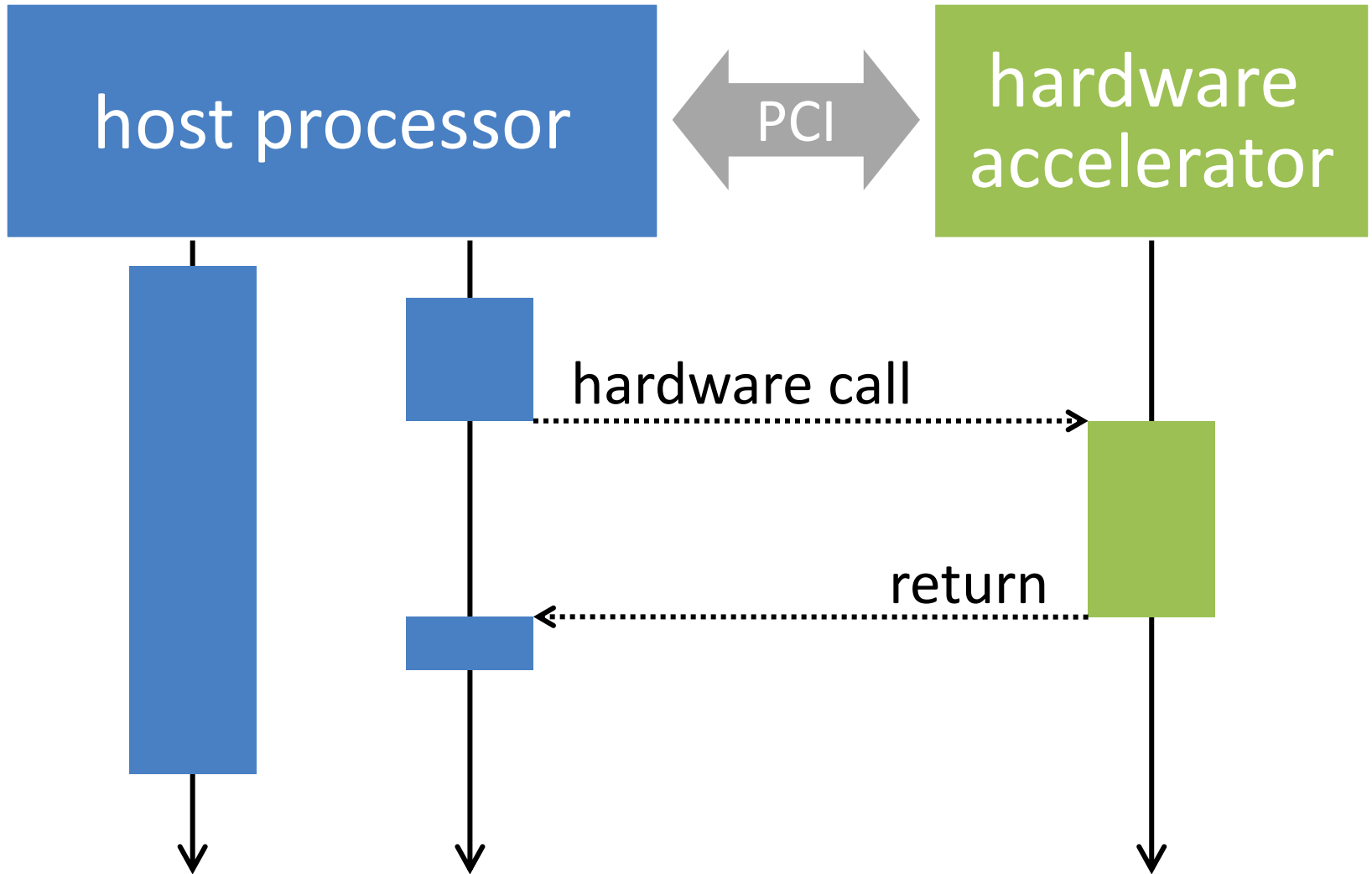
General-purpose processor and hardware accelerator form a hybrid system



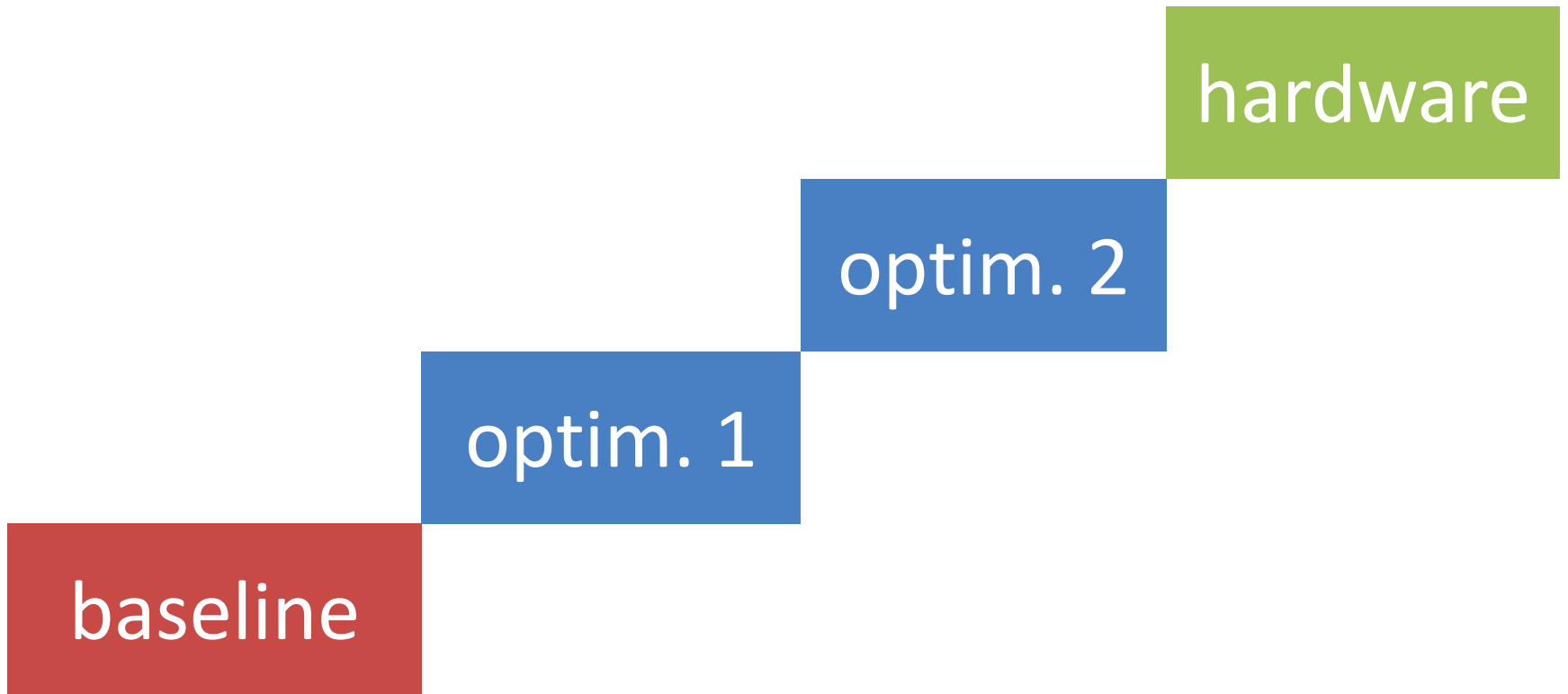
Java virtual machine acts as
an abstraction layer for this hybrid system



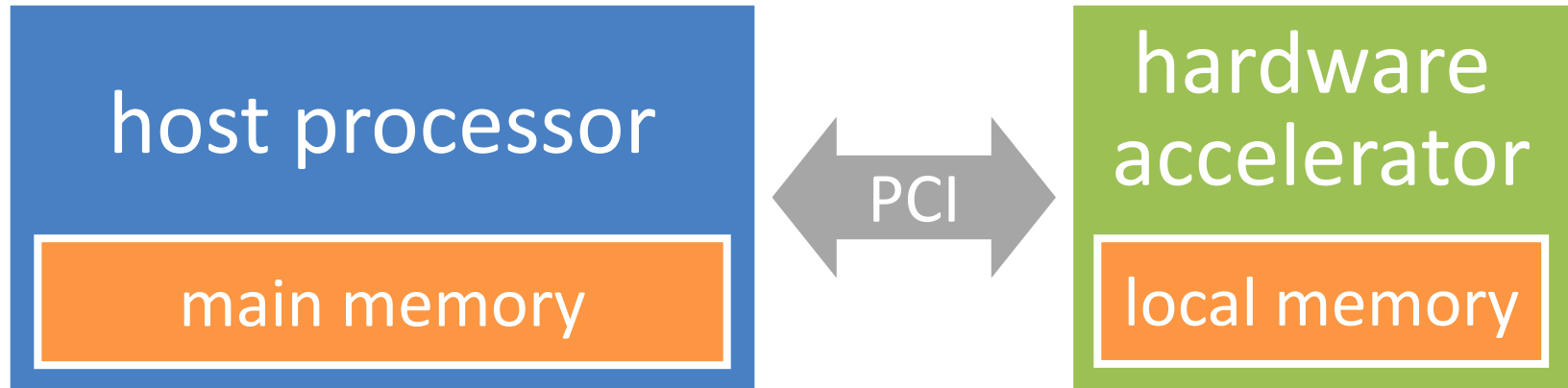
Virtual machine delegates control to the hardware accelerator



Virtual machine translates Java bytecode to native machine code ... or hardware?

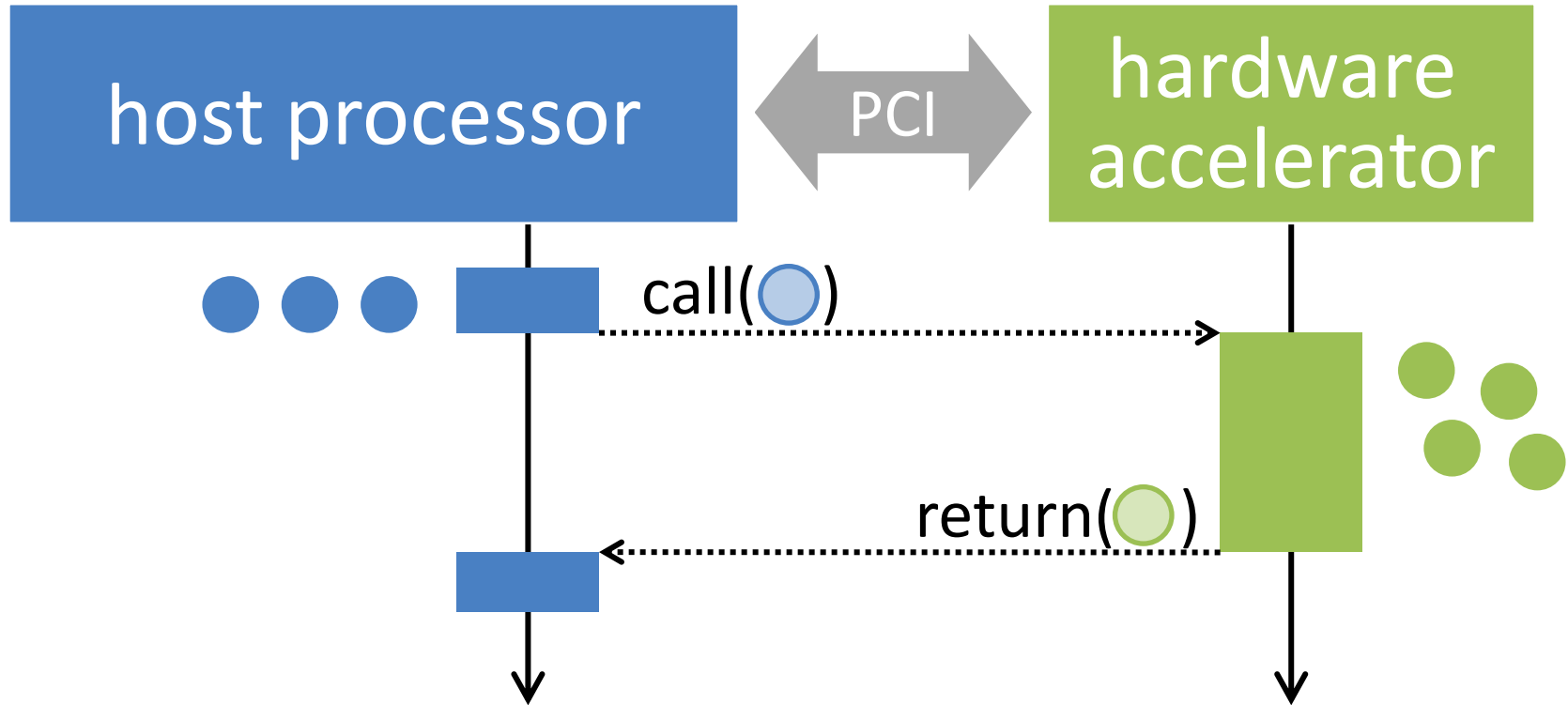


Hybrid architecture benefits from communication-aware memory allocation



- Distributed Java heap:
both components can access all objects
- NUMA: non-uniform memory access
- Optimal data allocation is very important

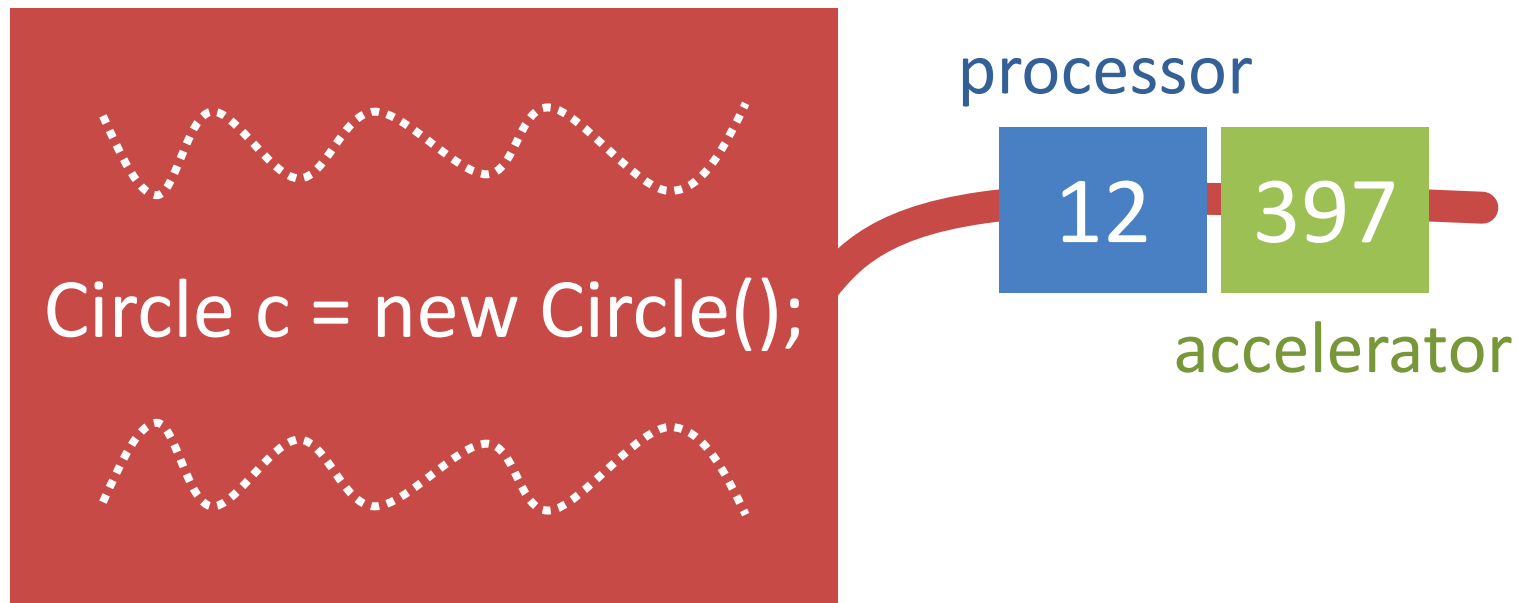
Local allocation: memory location is based on the creating component



- Each component creates objects in its own memory
- Best solution for most short living objects

Self-learning strategy tries to find the optimal memory allocation

- Count loads & stores per component
- New objects are allocated based on these counters
- Default: allocate in main memory



Determining which strategy performs best for each benchmark

- DaCapo and SPECjvm2008 benchmark suites
- Comparison of remote access ratio

47% baseline strategy

33% local allocation

26% self-learning

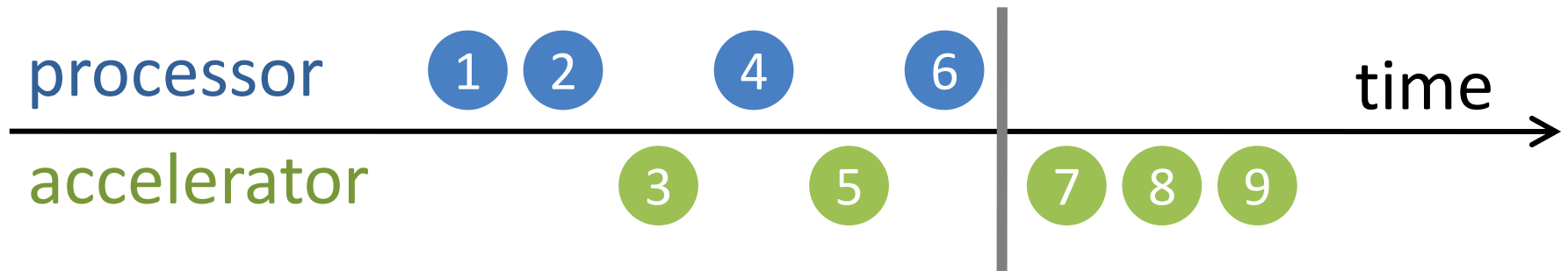
11% optimum

Self-learning strategy is a good choice for most benchmarks



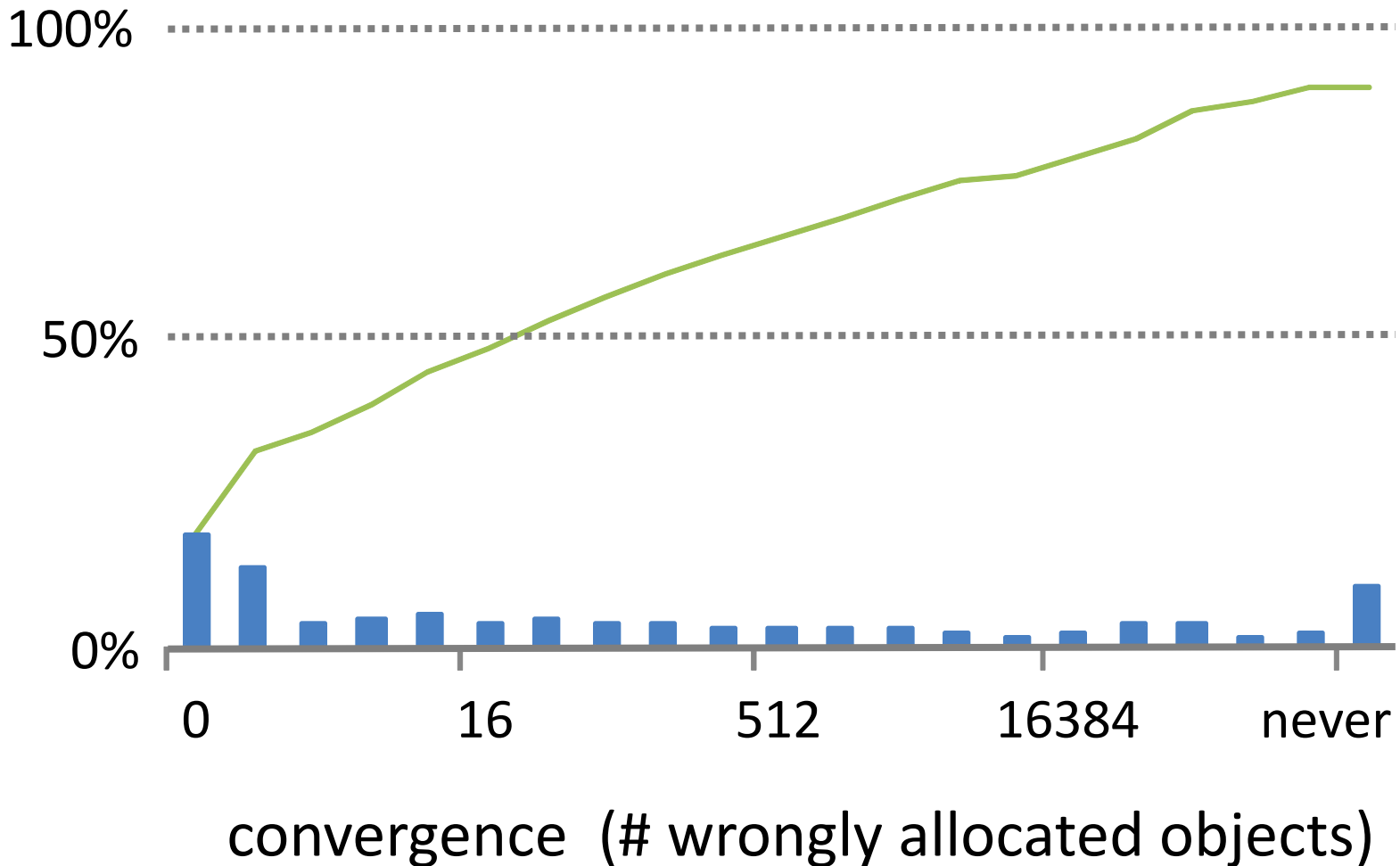
Most creation sites reach to a stable allocation policy after only a few objects

```
Circle c = new Circle();
```



- Except from the first 6 objects, all objects are allocated in the 'optimal' memory

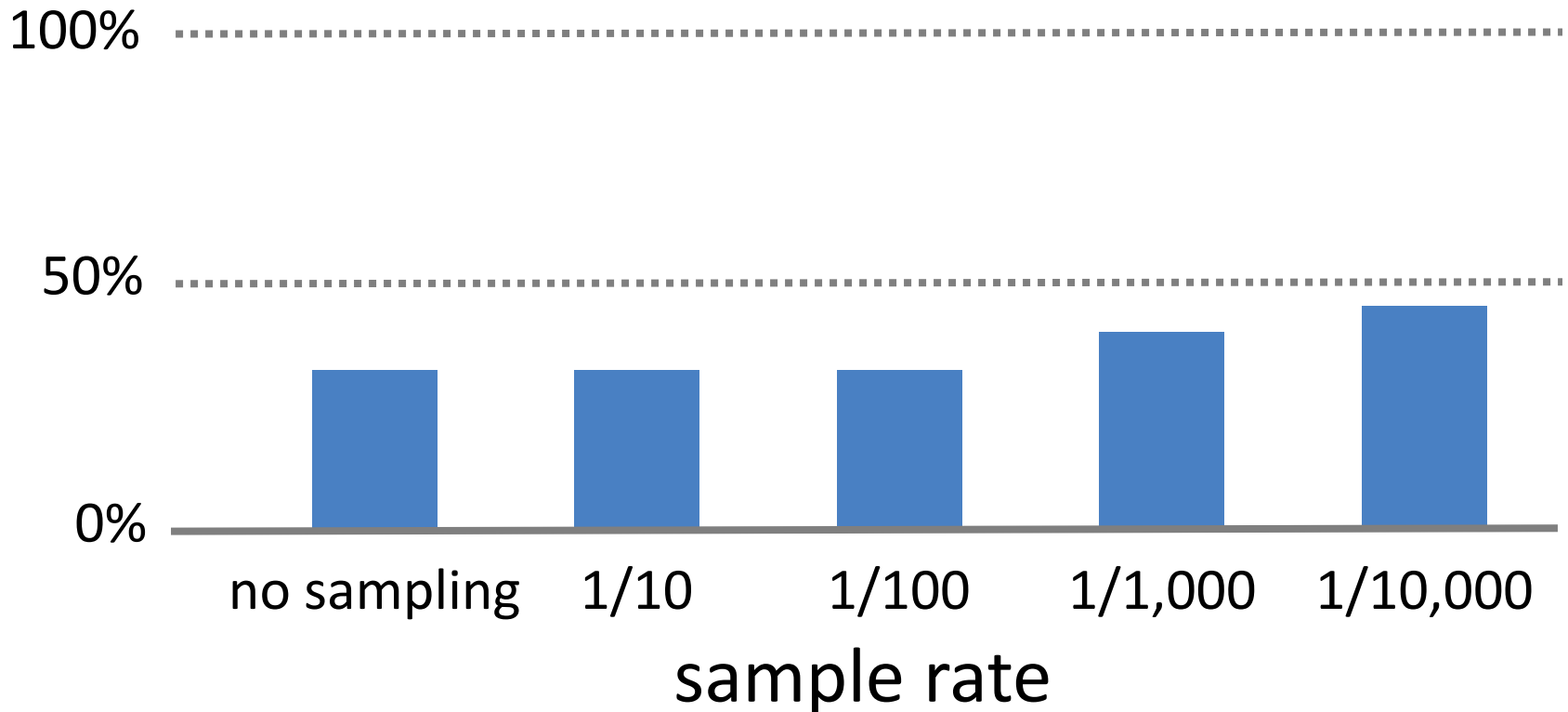
Some objects are placed before stabilisation, but self-learning algorithm learns very fast



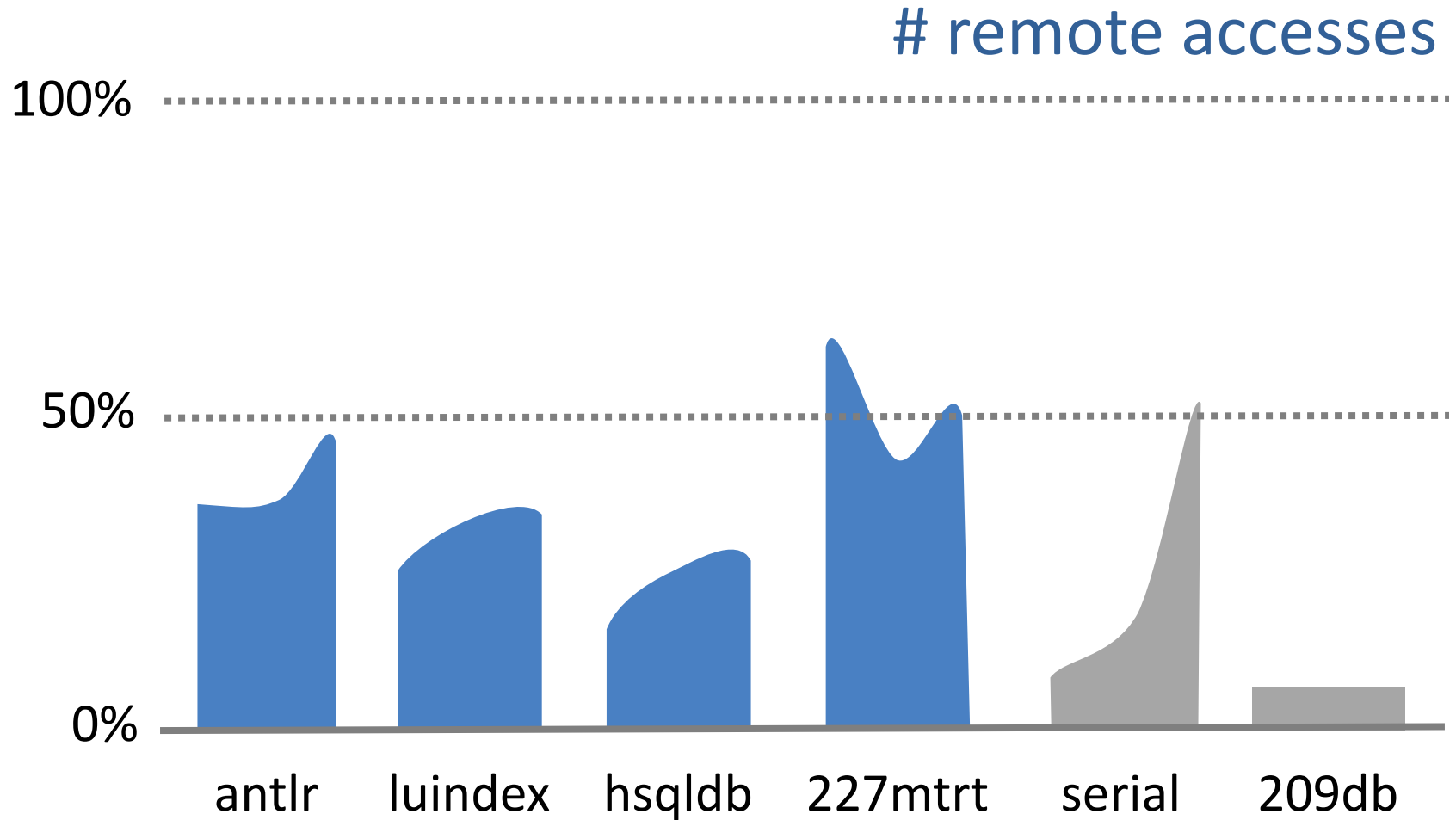
Sampling data usage patterns increases the remote access ratio

- Evaluation for benchmark antlr
- Limited performance degradation

remote accesses

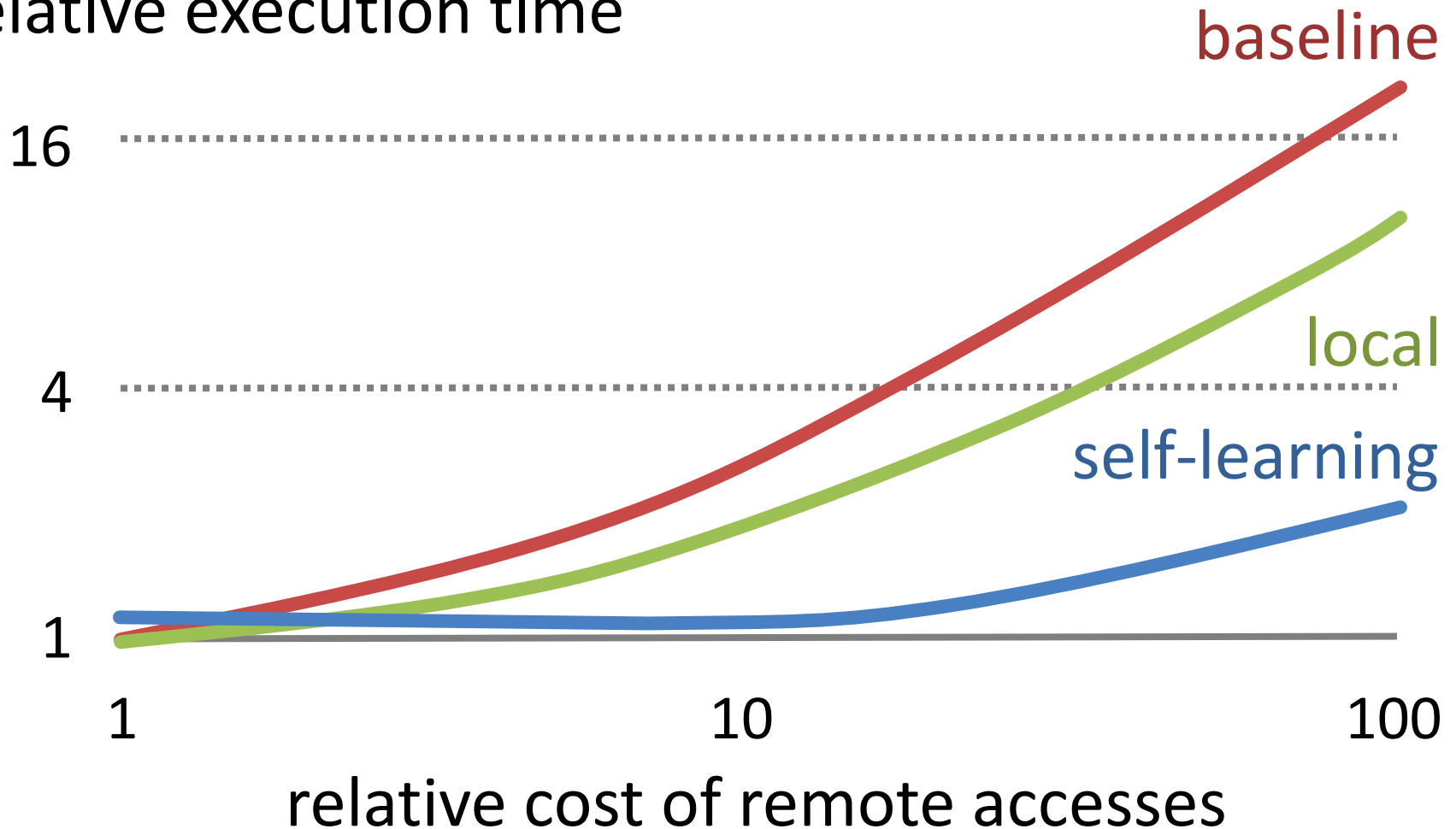


Self-learning strategy also works with sampled data usage patterns



More programs benefit from hardware-acceleration due to self-learning strategy

relative execution time



Hardware-accelerated JVM benefits from self-learning memory allocation

- Remote accesses are a problem
 - 47% of all memory accesses
 - Slow communication channel
- Self-learning memory allocation solves this problem
 - Significant reduction of remote accesses
 - Very fast convergence
 - Limited run-time overhead due to sampling