Titanium

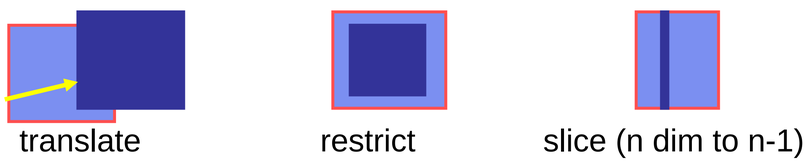
http//titanium.cs.berkeley.edu

Overview

- **Based on Java, an object-oriented language**
  - Classes, memory management, type safety
  - Compiled to C and then machine code
  - No Java virtual machine
- **Parallelism**
  - SPMD parallelism like CAF and UPC
  - Partitioned global address space
  - Checked, deadlock-free barriers

Features Added to Java

- **Multidimensional arrays**
  - Iterators, subarrays, transpose, copying
  - Rich calculus on indices without copying data

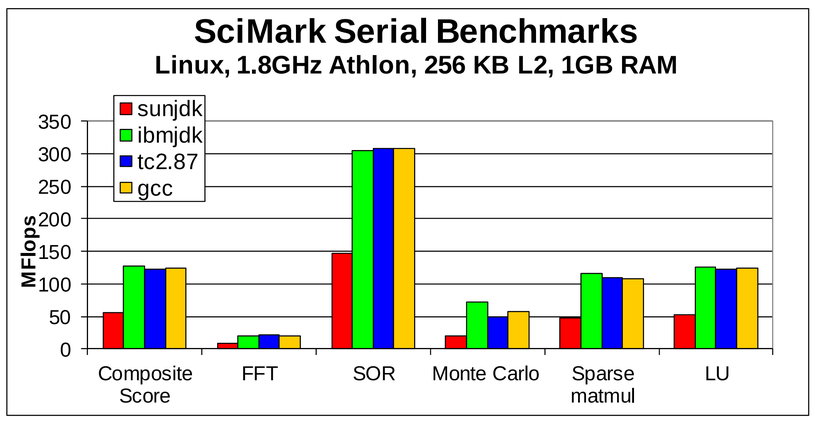



```
RectDomain<2> r = [0:100,100];
double [2d] a = new double [r];
foreach(p in a.domain().shrink(1))
   a[p] = 2*a[p];  // double interior values
```

- **Type system extensions**
  - Immutable classes for complex numbers, etc.
  - Templates and operator overloading
  - Local/global qualifier on references
  - Single-valued (replicated) variables
- **Zone-based memory management**
  - User managed, mixed with garbage collection
- **Libraries**
  - Collective communication, bulk I/O, profiling, serialization for checkpointing

Serial Performance

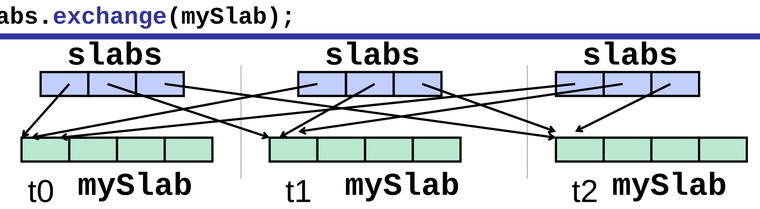
- **Compiler optimizes loops, arrays, inheritance,…**
- **Serial performance comparable to C**



Distributed Data Types

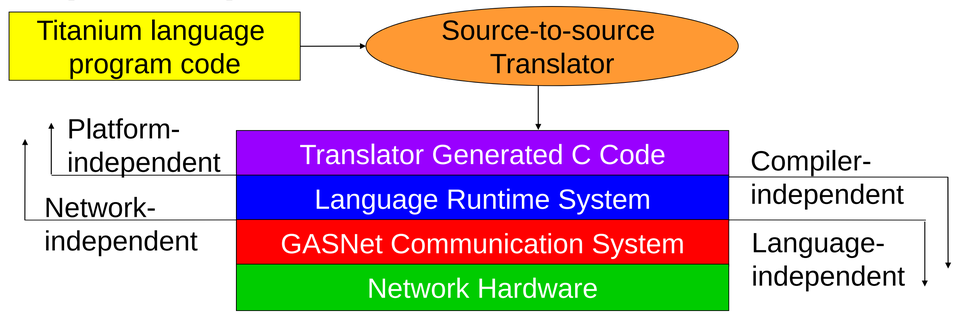
- **Use of pointer-based directories**
- **Very general, distribution under user control**

```
double [3d] mySlab = new double [startCell:endCell];
// “slabs” array is pointer-based directory over all procs
double [1d] single [3d] slabs =
   new double [0:Ti.numProcs()-1] single [3d];
slabs.exchange(mySlab);
```



Titanium Compiler

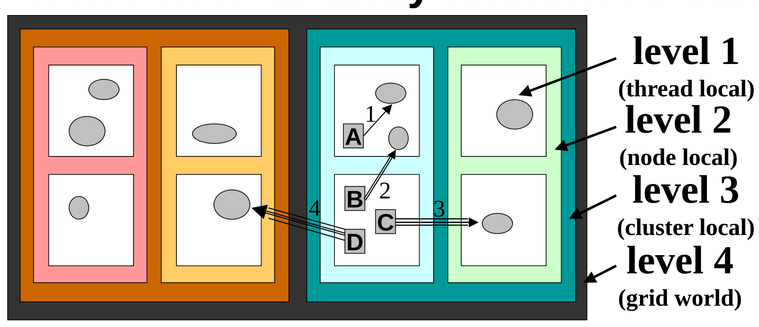
- **Open source implementation**
  - Runs on laptops, desktops, SMPs, clusters, supercomputers



- **Portable Titanium→C translator**
  - Uses GASNet for communication
  - Pthreads for shared memory

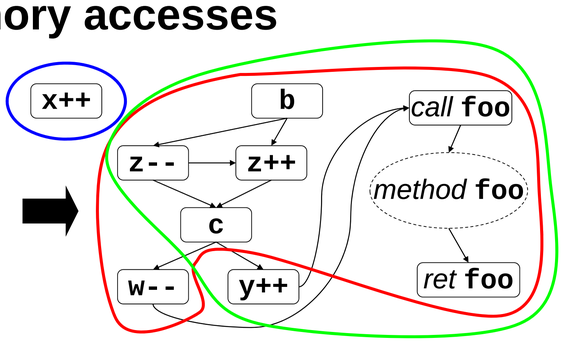
Analyses and Optimizations

- **Hierarchical PGAS pointer analysis**
  - Determines how far away referenced data may be

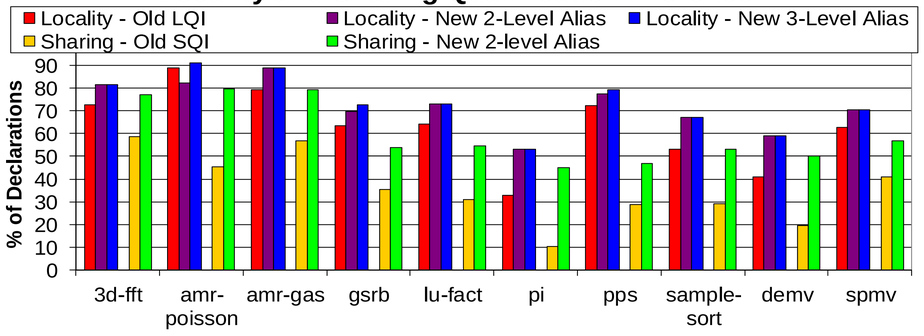


- **SPMD concurrency analysis**
  - Relies on SPMD model to determine potentially concurrent memory accesses

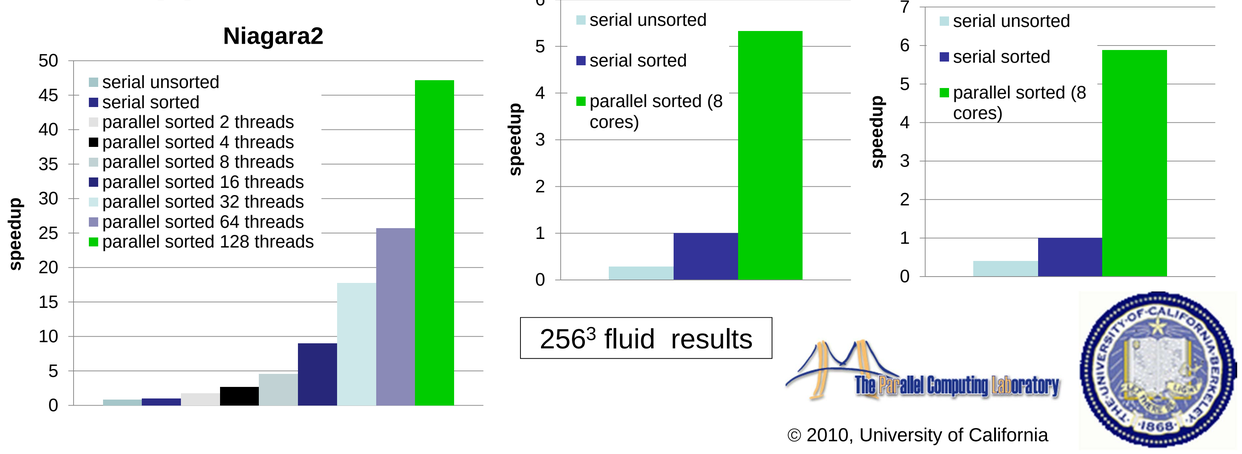
```
x++;
Ti.barrier();
if (b) z--;
else z++;
if (c [single]) w--;
else y++;
foo();
```



- **Analyses used to enforce sequential consistency and infer data locality and sharing**



- **Optimization of irregular updates on modern multicore architectures**
  - Applied to spread force operation from Heart application



256^3 fluid results

The Parallel Computing Laboratory