Explicit vs. Implicit Parallel Programming Language, Directive, Library

- Expose, Express, Exploit parallelism, synchronization, locality
- instruction-level parallelism (warm-up)
  - superscalar control unit
    - exposed in instruction reorder unit
    - expressed using register renaming
    - exploited in multiple instruction issue/execute/retire
  - VLIW control unit
    - exposed by compiler (unrolling, scheduling)
    - expressed in VLIW instructions
    - exploited by parallel operation issue
  - locality in register file
  - synchronization managed by reorder unit or by stalling

```
for( i = 0; i < n; ++i ) a[i] = b[i+1] + c[i+2];
```

Explicit vs. Implicit Parallel Programming Language, Directive, Library

- Expose, Express, Exploit parallelism; synchronization, locality
- vector parallelism (warm-up 2)
  - vector language extensions
    - exposed by application programmer
    - expressed in language extensions; remember Q8 functions?
    - exploited by parallel/pipelined functional units

```
a(1;n) = b(2;n) + c(3;n)
```

  - vectorizing compilers
    - exposed by application programmer (and compiler?)
    - expressed in vectorizable loops
    - exploited by parallel/pipelined functional units
  - locality in vector register file, if available
  - synchronization managed by hardware or compiler

```
do i = 1,n ; a(i) = b(i+1) + c(i+2) ; enddo
```

Scalable Parallelism – Node Level

- **MPI**
  - exposed in SPMD model
  - expressed in single program (redundant execution)
  - exploited one MPI rank per core
  - static parallelism can decompose based on MPI rank
  - send/receive exposes locality
  - sync implicit with data transfer
- **CAF (PGAS)**
  - exposed in SPMD model
  - expressed using single program (redundant execution)
  - one image per core
  - static parallelism can decompose, less general
  - get/put exposes locality
  - sync, separate from data transfer
- **HPF**
  - exposed in SPDD model
  - expressed using single program (implicitly executed redundantly)
  - one HPF processor per core
  - static parallelism (data parallel only)
  - load/store, locality hidden
  - synchronization mostly implicit managed by compiler

Shared Memory Parallelism – Socket/Core Level

- **Posix Threads**
  - exposed in application threads
  - expressed using pthread_create()
  - exploited one thread per core
  - dynamic parallelism, SPMD or not can compose
  - shared memory, coherent caches
  - sync using spin wait, more calls
- **Cilk**
  - exposed in asynchronous procedures
  - expressed using cilk_spawn
  - pool of threads, work stealing
  - dynamic parallelism can compose
  - shared memory, coherent caches
  - spin wait sync, or barriers
- **OpenMP**
  - expose in parallel loops, tasks
  - express with directives
  - one OpenMP thread per core
  - static parallelism (mostly) does support dynamic tasking can compose, nested parallelism
  - shared memory, coherent caches
  - barriers, task wait, ordered regions

Accelerator Parallelism – GPUs, etc.

- no library equivalent
- CUDA or OpenCL
  - exposed in kernel procedures
  - expressed in CUDA kernels
    kernel domain, launch
  - grid parallelism
    thread block parallelism
    accelerator asynchronous with host
  - static parallelism, does not compose
  - sync explicit within thread block
  - sync implicit between kernels
  - exposed memory hierarchy
    host, device, sw cache, register
- PGI Accelerator Model
  - exposed in nested parallel loops
  - expressed in nested parallel loops,
    accelerator directives
  - exploited as above
  - static parallelism, data parallel only
    does not compose
  - limited synchronization
  - locality managed by compiler

Abstraction Levels

- Library
  - independent of compiler
  - opaque to compiler
- Language
  - allows optimization
  - requires compiler
- Directives
  - allows optimization
  - requires compiler
  - may preserve portability
  - may allow specialization
- Node Level
  - scalable, static parallelism
  - emphasis on locality
- Socket/Core Level
  - static+dynamic parallelism
  - locality unaddressed
  - cache coherence
- Accelerators
  - regular parallelism
  - locality exposed