**Introduction**

Is there a simple and succinct mathematical structure underlying all three forms of parallelism: clustering, multicore and manycore?

To what extent can we lift the abstraction level of programming without sacrificing performance?

---

**Basic Idea:** representing both data and threads as hierarchical array types that form an algebraic system

\[
A \triangleq \text{pinned float}[[3][2]][4] \quad B \triangleq \text{dmem float}[[\# A^01][\# A^00]][\# A^1]
\]

- A data array type contains information about its memory location;
- A thread array type contains information about its threads;
- Types may refer to each other.

---

**Additional Feature:** Single Program Multiple Codeblocks

```c
#detour <mpi_thread_array_type> {
    mpi_code
    #detour <pthread_array_type> {
        CPU_multi_thread_code
        #detour <pthread_array_type> {
            GPU_kernel_code
            CPU_multi_thread_code
        }
    }
    mpi_code
}
```

- 20-line deeply optimized FFT code for GPU clusters (Tianhe-1A)

---

**Case Study:** direct simulation of turbulent flows (300-line par. code)

- 4096 3D completed, 8192 3D work-in-progress, 14336 3D tested.

---

**Various Data Transfer Patterns Unified**

\[
B \leftarrow A 
\]

for (. . ) cudaMemcpy(H2D);

\[
[[ \# D^0 ][\# T]][\# D^1] \leftarrow [[\# T][\# D^0]][\# D^1] \quad \text{MPI_Alltoall}
\]

\[
[\# T][\# D^1] \leftarrow [[1 \# T][\# D^0]][\# D^1] \quad \text{MPI_Scatter}
\]

---

**Other Interests:** Imperative, Object-Oriented, Probabilistic and Pointer Programming Theories.