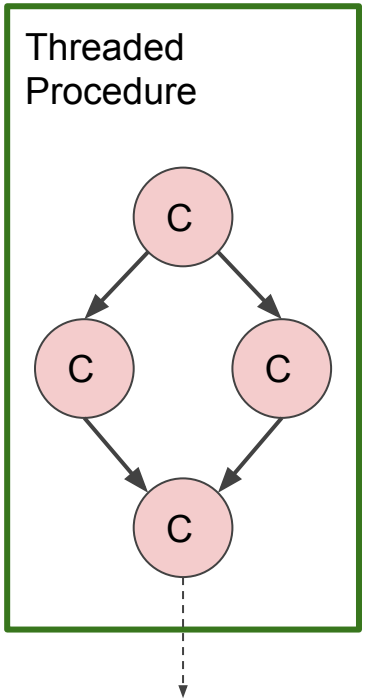
CAPSL Research Group

UDEL GPU Hackathon

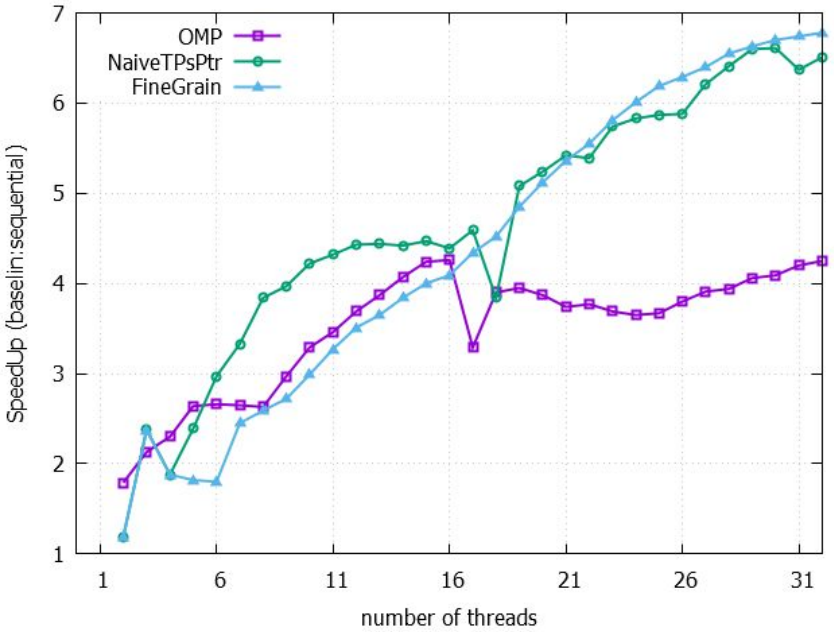
INTRODUCTION

- Some scientific applications feature heavy dependence patterns.
  - Examples: LU decomposition, Cholesky factorization
  - We'll focus on stencils in the rest of this presentation
- Most implementations rely on coarse grain synchronization
- We want to use finer-grain synchronization—Codelet Model
  - Fine grain data-driven program execution model.
  - Tasks decomposed in lightweight non-preemptive tasks.
  - Two levels of parallelism.



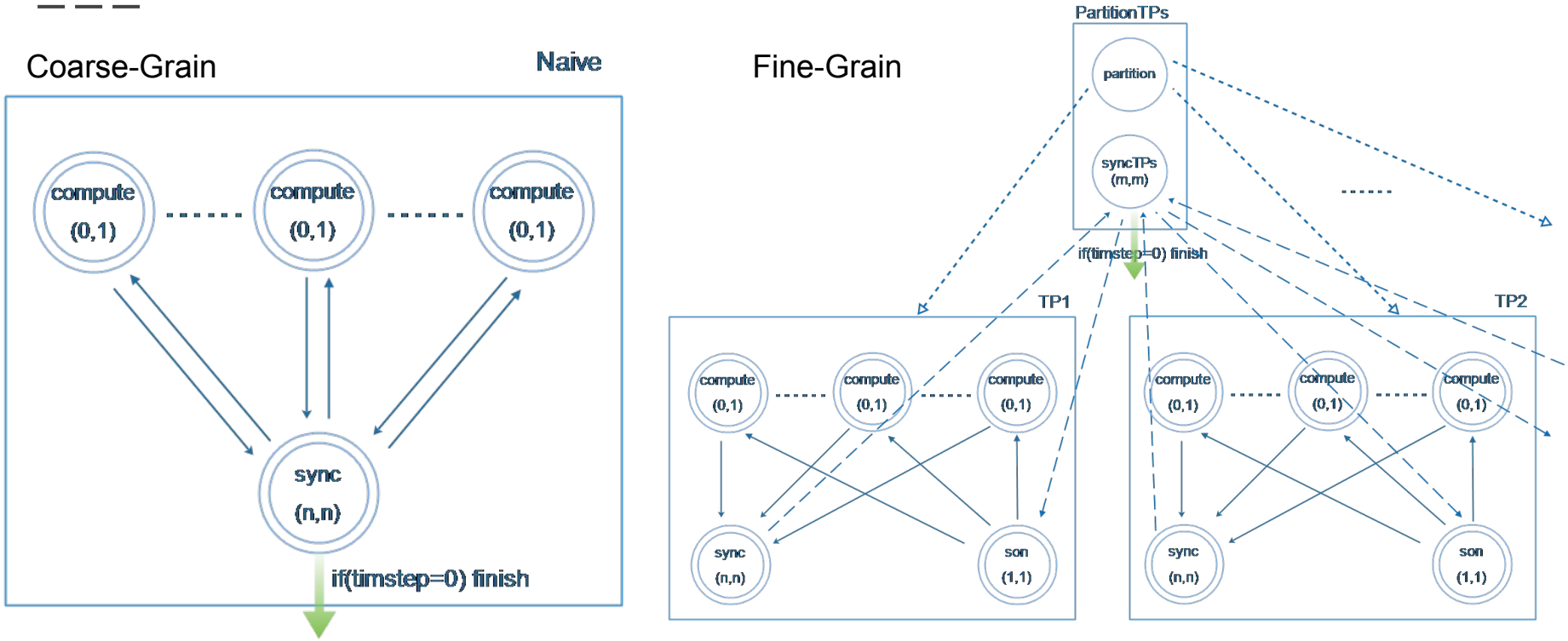
Stencil computation: From Coarse Grain to Fine Grain

```
void  stencil_5pt(
    double* restrict dst,    double* restrict src,
    const size_t     n_rows, const size_t     n_cols,
    size_t           n_steps)
{
  typedef double (*Array2D)[n_cols];
# pragma omp parallel default(none) shared(src, dst) \
            firstprivate(n_rows, n_cols, n_tsteps)
  {
    Array2D  DST = (Array2D) dst,
             SRC = (Array2D) src;
    size_t n_ts  = n_tsteps;
    while (n_ts-- > 0) {
#     pragma omp for nowait
      for (size_t i=1; i<n_rows-1; ++i)
        for (size_t j=1; j<n_cols-1; ++j)
          DST[i][j] = 0.25 * (SRC[i-1][j]+SRC[i+1][j]
                            + SRC[i][j-1]+SRC[i][j+1]);
      SWAP_PTR(&DST,&SRC);
#     pragma omp barrier
    }
  }
}
```



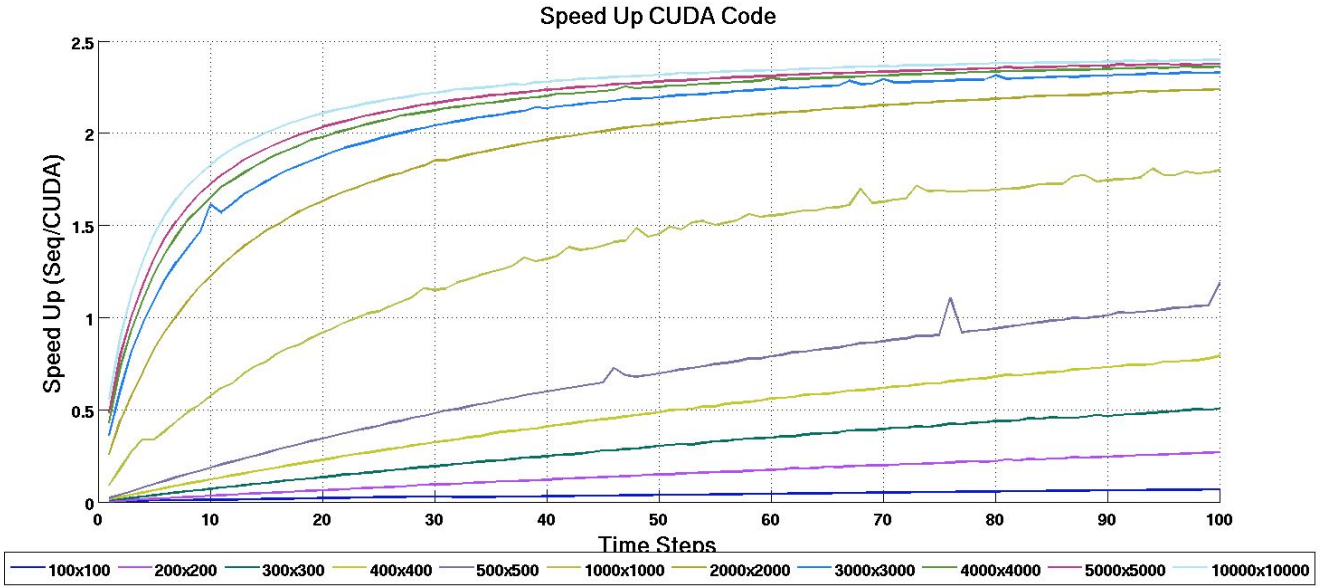
Strong Scaling. 3000x3000 2D 5 points Stencil
2 Sockets Intel Sandy Bridge (32 PEs) 20MiB LLC

From Coarse-Grain to Fine-Grain Stencil Computation



Current GPU Implementation CUDA

- 10x10 Tiling. 5x5 Threads, N/5×N/5 Blocks.
- Low ~2.5x Speed up for big matrix sizes vs ~50x pure GPU code



Our Goal: Heterogeneous Computing: CPU+GPU in a Data-Driven Environment

1. Optimize GPU kernel to better utilize the hardware
2. Potential optimizations include:
   - double-buffering,
   - separating the computation relying on ghost cells and the truly independent computations,
   - etc.
3. Include the GPU kernel within our CPU-based implementation
4. Evaluate the overall resulting speedup
5. **Goal: Optimize the GPU kernel to correctly overlap with host↔GPU transfers**