Seismic Wave Propagation on Heterogeneous Systems with Chapel

Alexey Gokhberg and Andreas Fichtner

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

CSCS
Swiss National Supercomputing Centre

Motivation

Hardware

- CSCS Cray XC30 "Piz Daint"
- 5,272 computing nodes:
  - 8-core Intel Xeon E5-2670 CPU, 32 GB RAM
  - NVIDIA Tesla K20X, 6 GB GDDR5

Programming

- Fortran, C, C++
- MPI
- CUDA, OpenACC

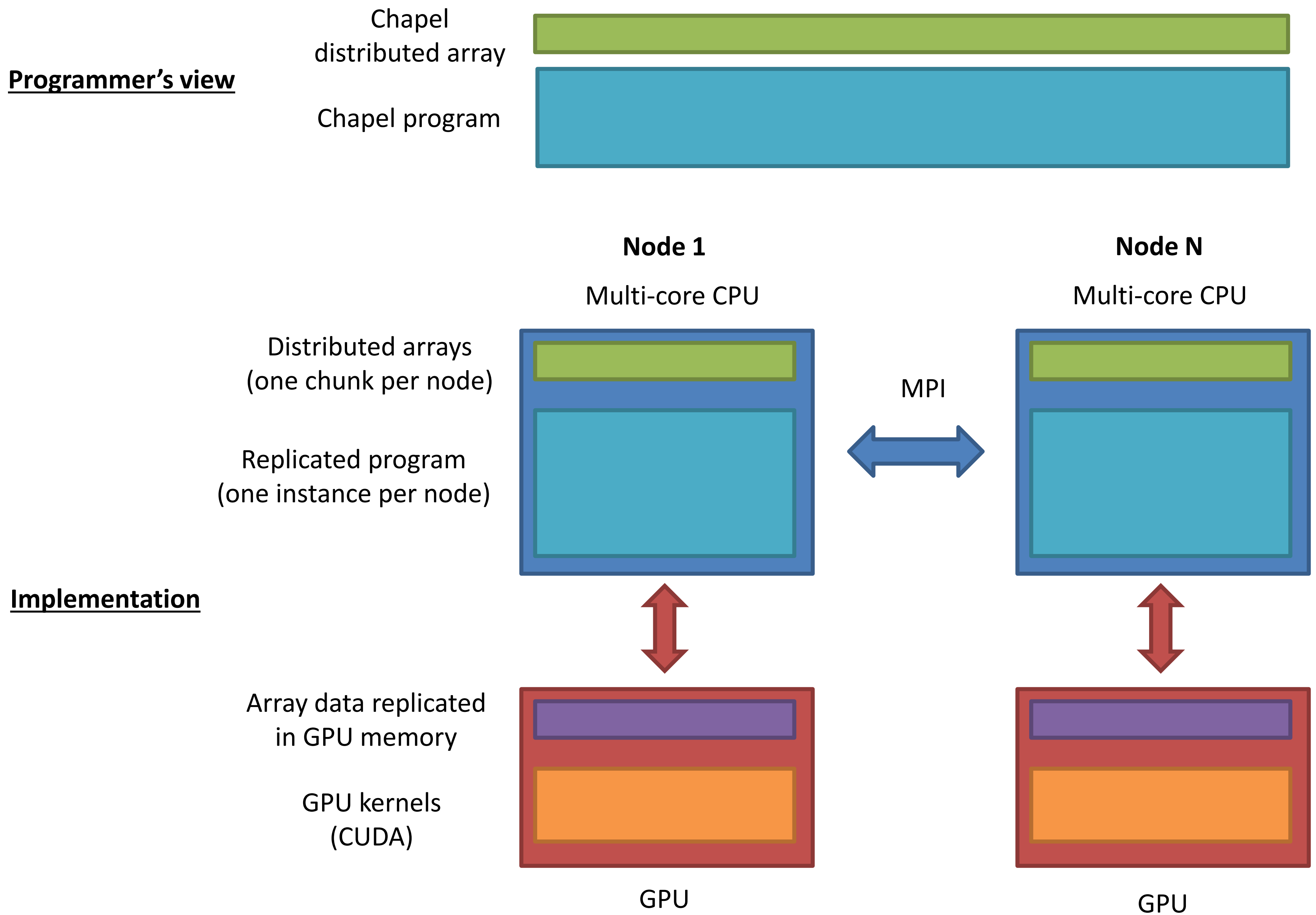
Challenges

- a gap between hardware complexity and programmer productivity
- restricted portability of application code

Chapel

- emerging parallel programming language
- originally developed at Cray Inc.
- part of DARPA-led High Productivity Computing Systems (HPCS) program, 2003-2012
- designed for programmer productivity

Programming Model



Engineering Strategy

Cray:

- implement the complete (rather complex) language
- application performance is not a priority
- no GPU support
- still work in progress

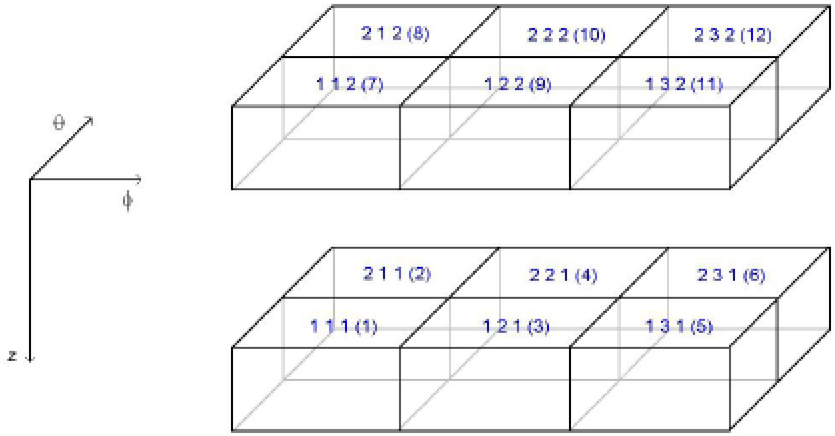
Our research:

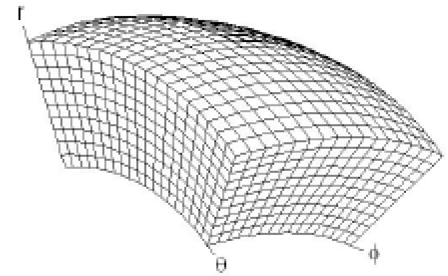
- implement a small yet representative language subset
- sufficient for programming a selected class of applications
- competitive application performance
- GPU support required

Methodology:

- select a reference application
- implement the reference application using conventional technologies:
    - homogeneous (CPU only)
    - heterogeneous (CPU and GPU)
- define and implement a subset of Chapel
- implement the reference application in Chapel
- assess code quality and application performance

Reference Application: SES3D

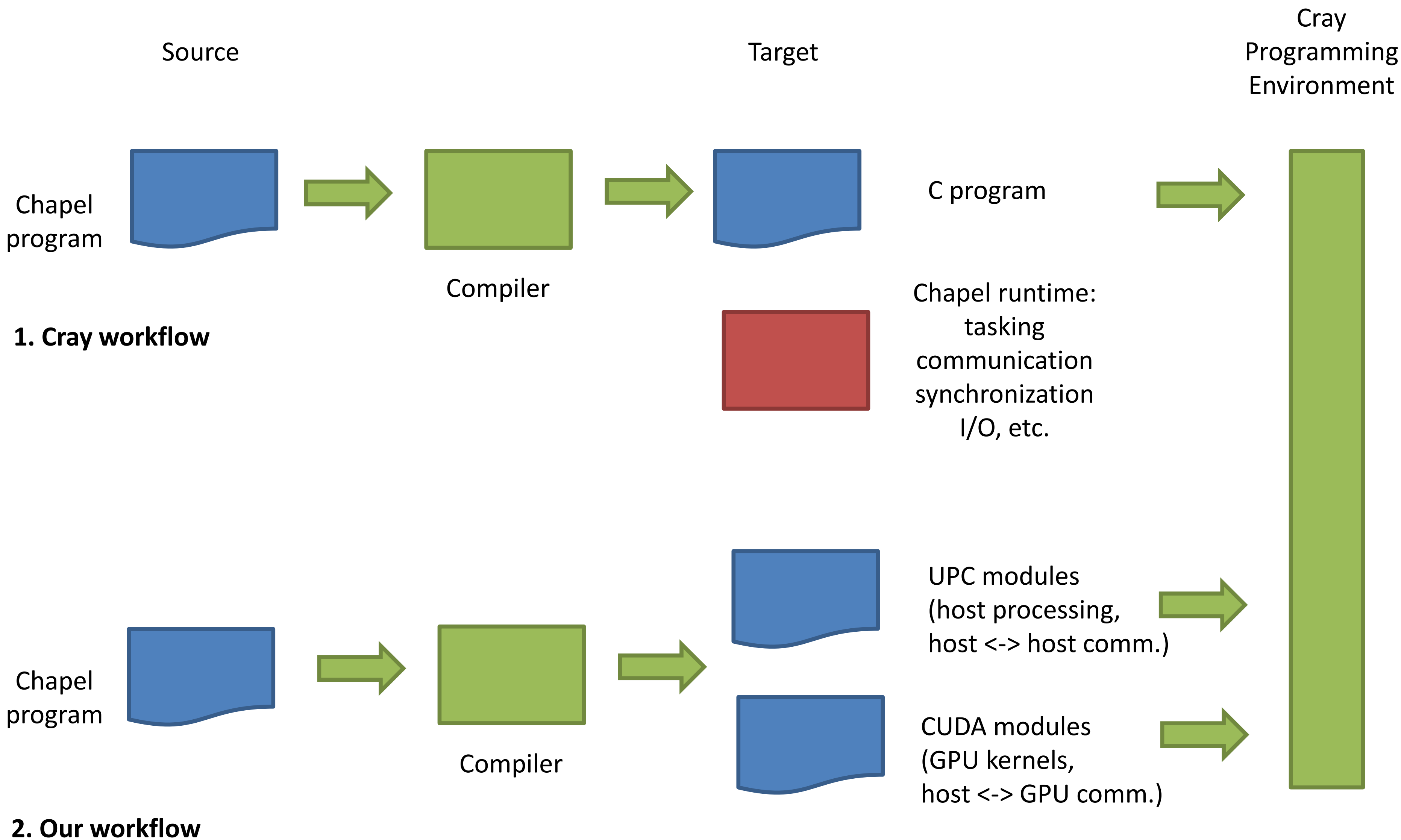




- elastic wave propagation and waveform inversion in a 3D spherical section
- based on a spectral-element discretization of the seismic wave equation combined with adjoint techniques
- implemented in Fortran 90 + MPI

- works on large data domain
- has enough inherent data parallelism
- performs mostly local computations

Implementing Chapel

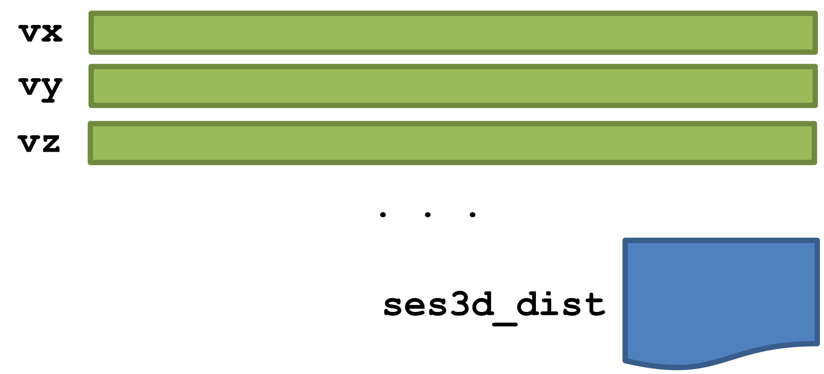


UPC = Unified Parallel C
(essentially C extended with the support for shared arrays)

Data Mapping

Chapel

```
var vx: [MD][ED] float;
var vy: [MD][ED] float;
var vz: [MD][ED] float;
. . .
```

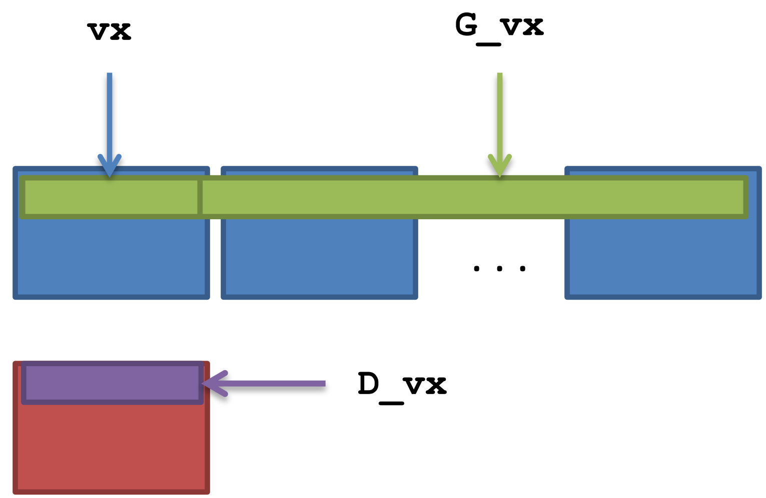


UPC

```
shared[MD_MAX_LOC*ED_MAX]
    float G_vx[THREADS*MD_MAX_LOC][ED_MAX];
float (*vx)[ED_MAX];
```

CUDA

```
__device__ float D_vx[MD_MAX_LOC][ED_MAX];
```



Computation Mapping (CPU)

Chapel

```
forall IM in MD do {
    forall IE in ED do {
        . . .
        vx[IM][IE] += dt * (sx[IM][IE] - ispml * prof[IM][IE] * vx[IM][IE]);
        . . .
        }
    }
```

UPC

```
for (IM = 0; IM < MD_length; IM++) {
    for (IE = 0; IE <= 255; IE++) {
        . . .
        vx[IM][IE] += dt * (sx[IM][IE] - ispml * prof[IM][IE] * vx[IM][IE]);
        . . .
        }
    }
```

Computation Mapping (GPU)

Chapel

```
forall IM in MD do {
    forall IE in ED do {
        . . .
        vx[IM][IE] += dt * (sx[IM][IE] - ispml * prof[IM][IE] * vx[IM][IE]);
        . . .
        }
    }
```

CUDA

```
__global__ void kernel03() {
    int idx = blockDim.x * blockIdx.x + threadIdx.x;
    if (idx >= D_MD_length * ED_MAX)
        return;
    int IM = idx / ED_MAX;
    int IE = idx % ED_MAX;
    . . .
    D_vx[IM][IE] += D_dt * (D_sx[IM][IE] - D_ispml * D_prof[IM][IE] * D_vx[IM][IE]);
    . . .
    }

extern "C" void wrapKernel03() {
    . . .    // copy data from host to GPU
    int TPB = 256;
    int BPG = (MD_length * ED_MAX + TPB - 1) / TPB;
    kernel03<<<BPG, TPB>>>();
    . . .    // copy data from GPU to host
    }
```

Conclusion

Performance results

- CSCS Cray XK-7 “Tödi”
- 3 computing nodes / 48 CPU cores
- SES3D seismic wave simulation, 4000 iterations
- wall clock time:
    - original (Fortran 90 + MPI): 60 min
    - reference (hand-written UPC): 25 min
    - Chapel (generated UPC): 35 min

Further research

- extend the supported Chapel subset
- optimize communication
- optimize synchronization
- implement support for parallel I/O operations
- enhance utilization of GPU kernels