Mitigating Communication Costs in Emerging Parallel Programming Languages

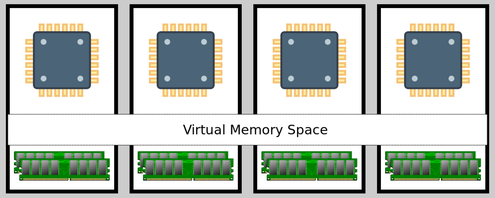
Engin Kayraklioglu
Advisor: Tarek El-Ghazawi

Parallel Programming Models

Shared Memory

No sense of data locality ⟶ Very easy to use

OS can map any virtual address to any physical address ⟶ Very inefficient



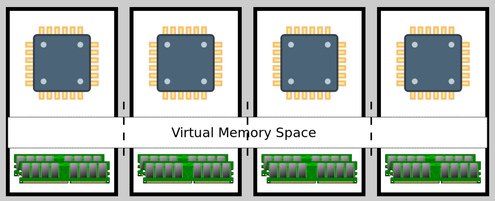
Message Passing

Data locality is enforced ⟶ Difficult to use

Programmer controls data traffic ⟶ High performance

Partitioned Global Address Space (PGAS)

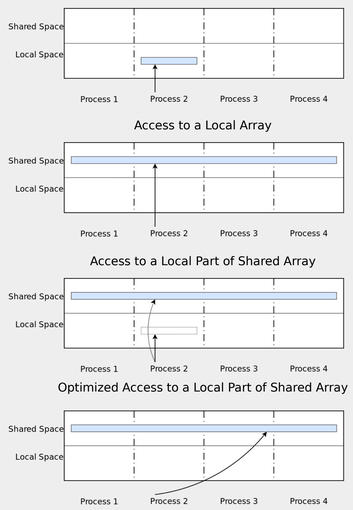
Flat memory ⟶ Easy to use
Language is locality-aware ⟶ Automatic optimizations
Programmer can exploit locality ⟶ Manual optimizations



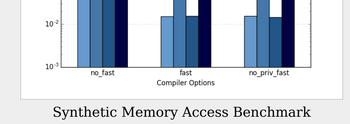
Stage 1
Performance Analysis[1]

Motivation

To analyze and compare different kinds of PGAS access performances in Chapel.



Access to a Local Array



Access to a Local Part of Shared Array

Optimized Access to a Local Part of Shared Array

Access to a Remote Part of Shared Array

Results

Access Type	Normalized Latency
Local	1x
Local Shared	35x
Local Shared Opt	1x
Remote	700x

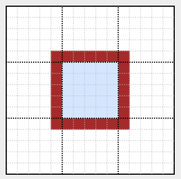
Synthetic Memory Access Benchmark

Stage 2
Hand Optimizations[2]

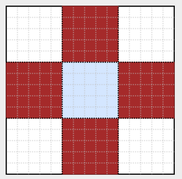
Motivation

What are some common practices for Chapel programmers to achieve better multi-node performance?

Optimizations



Sobel Access Pattern

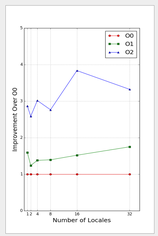


MM Access Pattern

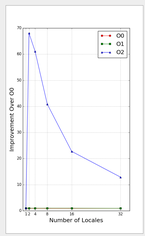
O0	No hand optimizations
O1	Reorganize loops to be able to use local block
O2	Manually localize remote data

Results

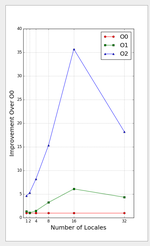
Version	Speedup	Lines of Code
O0	1x	1x
O1	1.1x-6.1x	4x-26x
O2	1.7x-68.1x	2.5x-11x



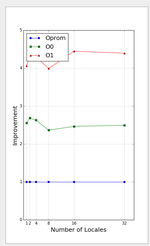
Sobel



MM



Heat diffusion



STREAM Triad

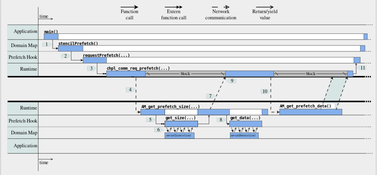
Stage 3
Runtime System Support*

Motivation

What if programming language supported user-driven prefetching?

Design

Productive
Extensible
Scalable
Efficient
Consistent



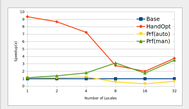
Results

Promising performance improvements with very little modification on application code

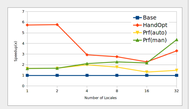
- Similar performance to manual prefetching
- Almost all experiments required adding few calls to prefetch functions

Ongoing Work

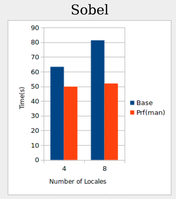
- Fixing some known implementation issues that are causing low performance in some cases
- Extending support to prefetching sparse data(matrices and graphs)



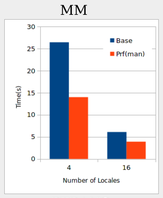
Sobel



MM



LULESH



PTRANS

What is Chapel?

Chapel is an emerging, productive, parallel programming language with the PGAS memory model[3]. Unlike other PGAS languages and libraries, Chapel is standalone and is designed from scratch for parallel programming.
Being a PGAS language, Chapel carries some inherent overheads of the easy-to-use memory model.

Distribution objects

```
var dom = {0..10} dmapped Block();
var arr: [dom] int;  //Block-distributed array
```

Data-parallel loops

```
// parallel-distributed loop
forall a in arr do
    something(a);
```

Task-parallel loops

```
// task parallel loop
coforall t in numTasks do
    something()
```

Tasks & Synchronization

```
var done$: sync bool;
begin {
    something();
    done$ = true;
}
// implicit wat
if done then
    somethingElse();
```

Locality control

```
// move execution to where
// arr[15] is
on arr[15] do
    local do something(arr[15]);
```

```
const imgSpace = {0..#2048, 0..#2048};
const imgDom = imgSpace
        dmapped Block(imgSpace);        // block distributed domain
var inImg: [imgDom] uint(8);            // block distributed array
var outImg: [imgDom] uint(8);           // block distributed array

// ... initialize img here ...

forall (i, j) in imgDom do              // distributed forall loop
  outImg = sobelKernel(inImg, i, j);    // run sobel kernel

-------------------------------------------------------------------
const matSpace = {0..#2048, 0..#2048};
const matDom = matSpace
        dmapped Block(matDom);          // block distributed domain
const A: [matDom] real;                 // block distributed array
const B: [matDom] real;                 // block distributed array
const C: [matDom] real;                 // block distributed array

// .. initialize matrices here ...

forall (i,j) in C.domain do             // distributed forall loop
  for k in A.domain.dim(2) do           // local-sequential for loop
    C[i,j] += A[i,k] * B[k,j];
```

References
[1]E. Kayraklioglu and T. El-Ghazawi, "Assessing Memory Access Performance of Chapel through Synthetic Benchmarks," in 2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), 2015, pp. 1147–1150.
[2]E. Kayraklioglu, O. Serres, A. Anbar, H. Elezabi, and T. El-Ghazawi, "PGAS Access Overhead Characterization in Chapel," in 2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), 2016, pp. 1568–1577
[3]chapel.cray.com

* This work is a collaboration with Chapel team at Cray: http://chapel.cray.com/collaborations.html

HPCL THE GEORGE WASHINGTON UNIVERSITY WASHINGTON DC
High Performance Computing Lab

engin@gwu.edu

THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC