LLVM-based Communication Optimizations for PGAS Programs

2nd Workshop on the LLVM Compiler Infrastructure in HPC @ SC15

Akihiro Hayashi (Rice University)
Jisheng Zhao (Rice University)
Michael Ferguson (Cray Inc.)
Vivek Sarkar (Rice University)

A Big Picture

X10,
Habanero-UPC++,...

Communication Optimization

©Berkeley Lab.

© Argonne National Lab.

© RIKEN AICS

Photo Credits : http://chapel.cray.com/logo.html, http://llvm.org/Logo.html, http://upc.lbl.gov/, http://commons.wikimedia.org/, http://cs.lbl.gov/

PGAS Languages

- High-productivity features:
  - Global-View
  - Task parallelism
  - Data Distribution
  - Synchronization

X10

Habanero-UPC++

CAF

Photo Credits : http://chapel.cray.com/logo.html, http://upc.lbl.gov/

Communication is implicit in some PGAS Programming Models

- Global Address Space
  - Compiler and Runtime is responsible for performing communications across nodes

Remote Data Access in Chapel

```
var x = 1;           // on Node 0
on Locales[1] {      // on Node 1
  … = x;             // DATA ACCESS
}
```

Communication is Implicit in some PGAS Programming Models (Cont'd)

Remote Data Access

```
var x = 1;          // on Node 0
on Locales[1] {// on Node 1
… = x;              // DATA ACCESS
```

Compiler Optimization

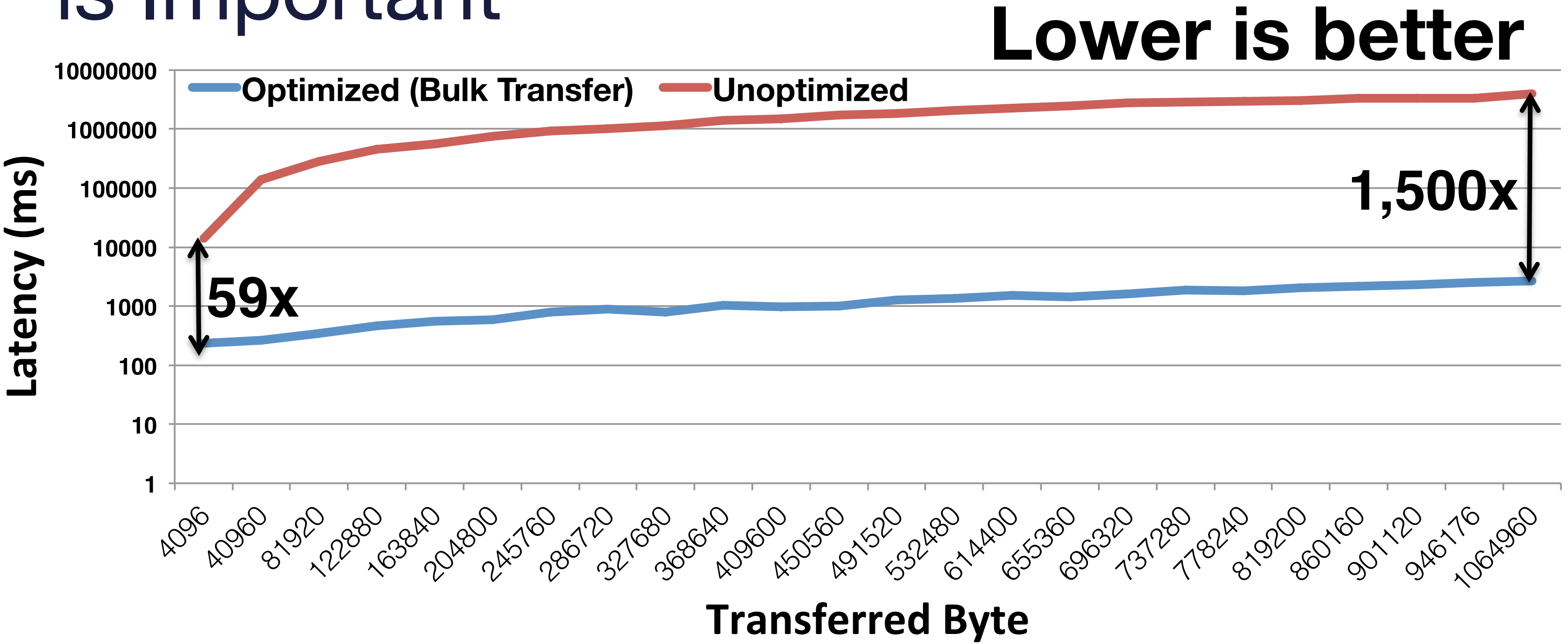
```
var x = 1;
on Locales[1] {
… = 1;
```

OR

Runtime affinity handling

```
if (x.locale == MYLOCALE) {
    *(x.addr) = 1;
else {
    gasnet_get(…);
}
```

Communication Optimization is Important



A synthetic Chapel program
on Intel Xeon CPU X5660 Clusters with QDR Inifiniband

PGAS Optimizations are language-specific

Chapel Compiler

Language Specific!

X10,
Habanero-UPC++,...

10 Compiler
Habanero-C Compiler

©Berkeley Lab.

© Argonne National Lab.

© RIKEN AICS

RICE

Photo Credits : http://chapel.cray.com/logo.html, http://llvm.org/Logo.html, http://upc.lbl.gov/, http://commons.wikimedia.org/, http://cs.lbl.gov/

Our goal

CRAY CHAPEL

BERKELEY UPC UNIFIED PARALLEL C

X10, Habanero-UPC++,…

LLVM

COMPILER INFRASTRUCTURE

©Berkeley Lab.

© Argonne National Lab.

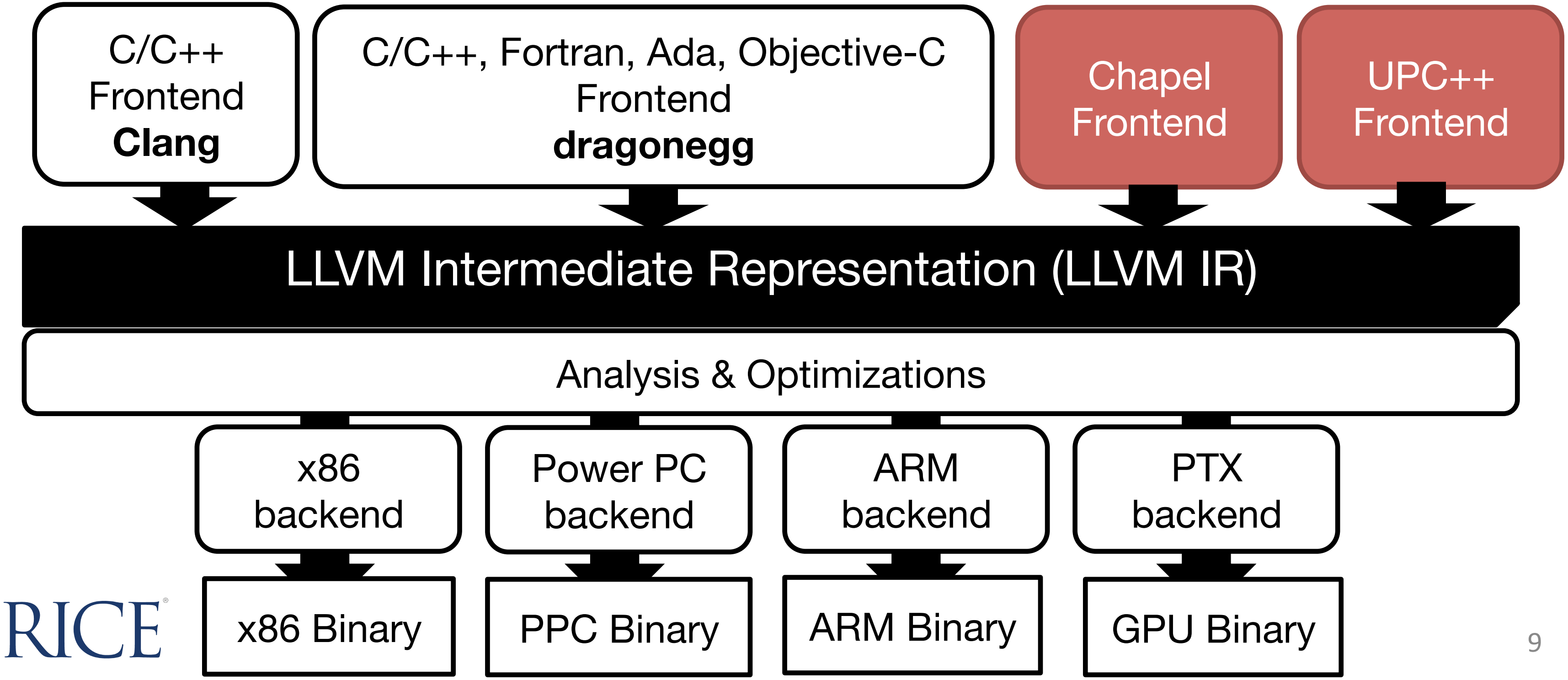
© RIKEN AICS

Photo Credits : http://chapel.cray.com/logo.html, http://llvm.org/Logo.html, http://upc.lbl.gov/, http://commons.wikimedia.org/, http://cs.lbl.gov/

RICE

Why LLVM?

- Widely used language-agnostic compiler

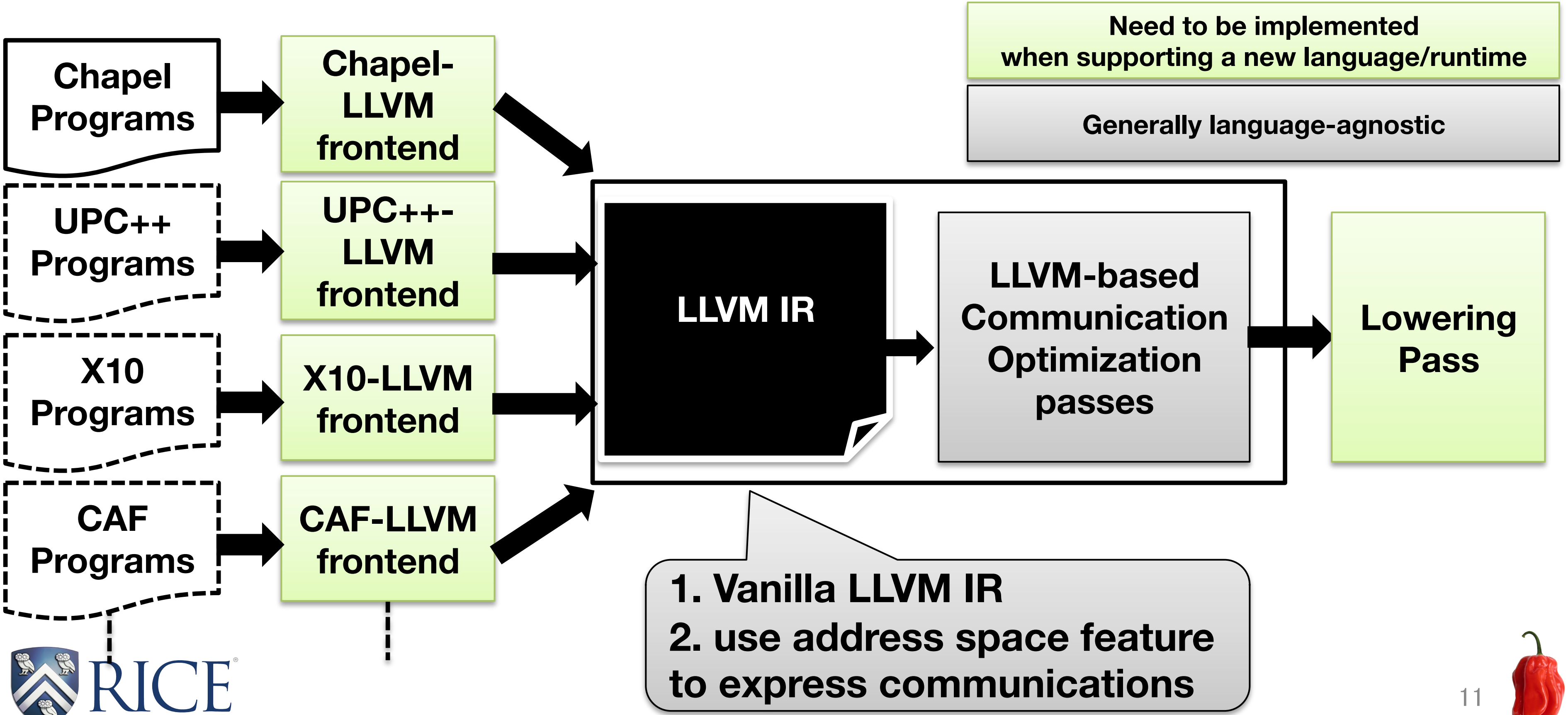


Summary & Contributions

- Our Observations :
  - Many PGAS languages share semantically similar constructs
  - PGAS Optimizations are language-specific
- Contributions:
  - Built a compilation framework that can uniformly optimize PGAS programs (Initial Focus : Communication)
    - ✓ Enabling existing LLVM passes for communication optimizations
    - ✓ PGAS-aware communication optimizations

Photo Credits : http://chapel.cray.com/logo.html, http://llvm.org/Logo.html,

Overview of our framework



How optimizations work

Chapel

```
// x is possibly remote
x = 1;
```

UPC++

```
shared_var<int> x;
x = 1;
```

```
store i64 1, i64 addrspace(100)* %x, …
```

treat remote access as if it were local access

1. Existing LLVM Optimizations
2. PGAS-aware Optimizations

Address space-aware Optimizations

Runtime-Specific Lowering

Communication API Calls

RICE

LLVM-based Communication Optimizations for Chapel

1. **Enabling Existing LLVM passes**
   - Loop invariant code motion (LICM)
   - Scalar replacement, …
2. **Aggregation**
   - Combine sequences of loads/stores on adjacent memory location into a single memcpy

These are already implemented in the standard Chapel compiler

An optimization example:
LICM for Communication Optimizations

LICM by LLVM

```
for i in 1..100 {
  %x = load i64 addrspace(100)* %xptr
  A(i) = %x;
}
```

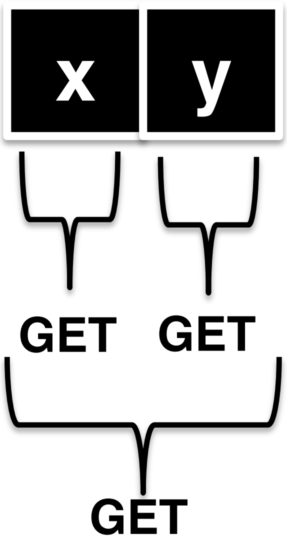
LICM = Loop Invariant Code Motion

An optimization example: Aggregation

```
// p is possibly remote
sum = p.x + p.y;
```

```
load i64 addrspace(100)* %pptr+0
load i64 addrspace(100)* %pptr+4
```

```
llvm.memcpy(…);
```



LLVM-based Communication Optimizations for Chapel

3. Locality Optimization

- Infer the locality of data and convert ***possibly-remote*** access to ***definitely-local*** access at compile-time if possible

4. Coalescing

- Remote array access vectorization

RICE

These are implemented, but not in the standard Chapel compiler

An Optimization example: Locality Optimization

```
proc habanero(ref x, ref y, ref z) {
    var p: int = 0;
    var A:[1..N] int;
    local { p = z; }
    z = A(0) + z;
}
```

1.A is definitely-local

2.p and z are definitely local

<u>3.Definitely-local access!</u>
<u>(avoid runtime affinity checking)</u>

An Optimization example: Coalescing

Before

```
1:for i in 1..N {
2:     … = A(i);
3:}
```

After

```
1:localA = A;
2:for i in 1..N {
3:     … = localA(i);
4:}
```

Perform bulk transfer

Converted to definitely-local access

Performance Evaluations: Benchmarks

Application	Size
Smith-Waterman	185,600 x 192,000
Cholesky Decomp	10,000 x 10,000
NPB EP	CLASS = D
Sobel	48,000 x 48,000
SSCA2 Kernel 4	SCALE = 16
Stream EP	2^30

Performance Evaluations: Platforms

- Cray XC30™ Supercomputer @ NERSC
  - Node
    - ✓ Intel Xeon E5-2695 @ 2.40GHz x 24 cores
    - ✓ 64GB of RAM
  - Interconnect
    - ✓ Cray Aries interconnect with Dragonfly topology
- Westmere Cluster @ Rice
  - Node
    - ✓ Intel Xeon CPU X5660 @ 2.80GHz x 12 cores
    - ✓ 48 GB of RAM
  - Interconnect
    - ✓ Quad-data rated infiniband

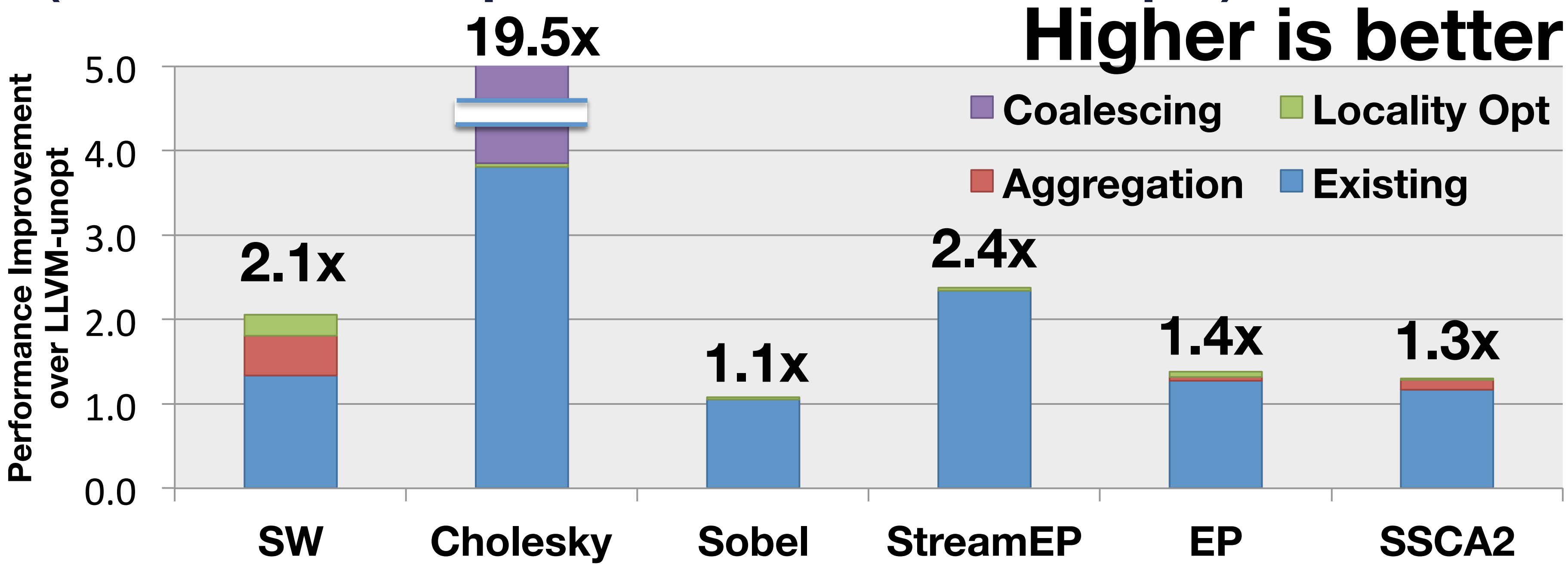
Performance Evaluations: Details of Compiler & Runtime

- Compiler
  - Chapel Compiler version 1.9.0
  - LLVM 3.3
- Runtime :
  - GASNet-1.22.0
    - ✓ Cray XC : aries
    - ✓ Westmere Cluster : ibv-conduit
  - Qthreads-1.10
    - ✓ Cray XC: 2 shepherds, 24 workers / shepherd
    - ✓ Westmere Cluster : 2 shepherds, 6 workers / shepherd

Performance Evaluation

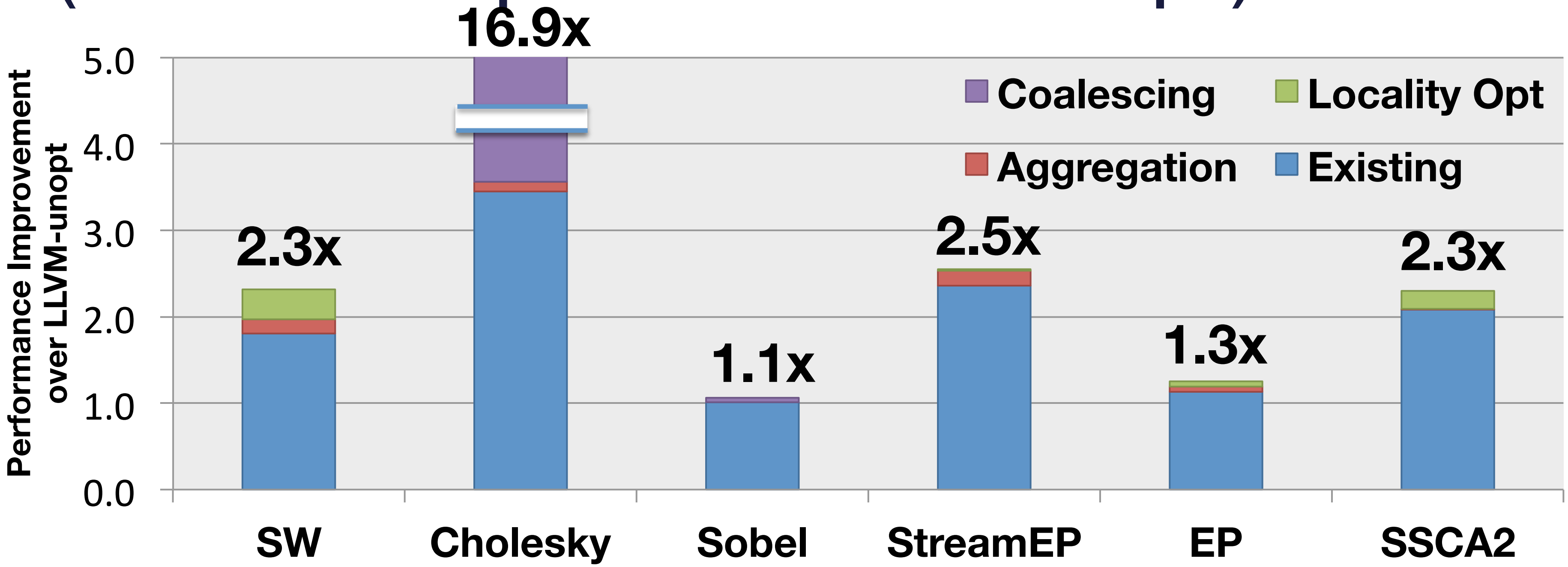
BRIEF SUMMARY OF PERFORMANCE EVALUATIONS

Results on the Cray XC (LLVM-unopt vs. LLVM-allopt)



✓ *4.6x performance improvement relative to LLVM-unopt on the same # of locales on average (1, 2, 4, 8, 16, 32, 64 locales)*

Results on Westmere Cluster (LLVM-unopt vs. LLVM-allopt)

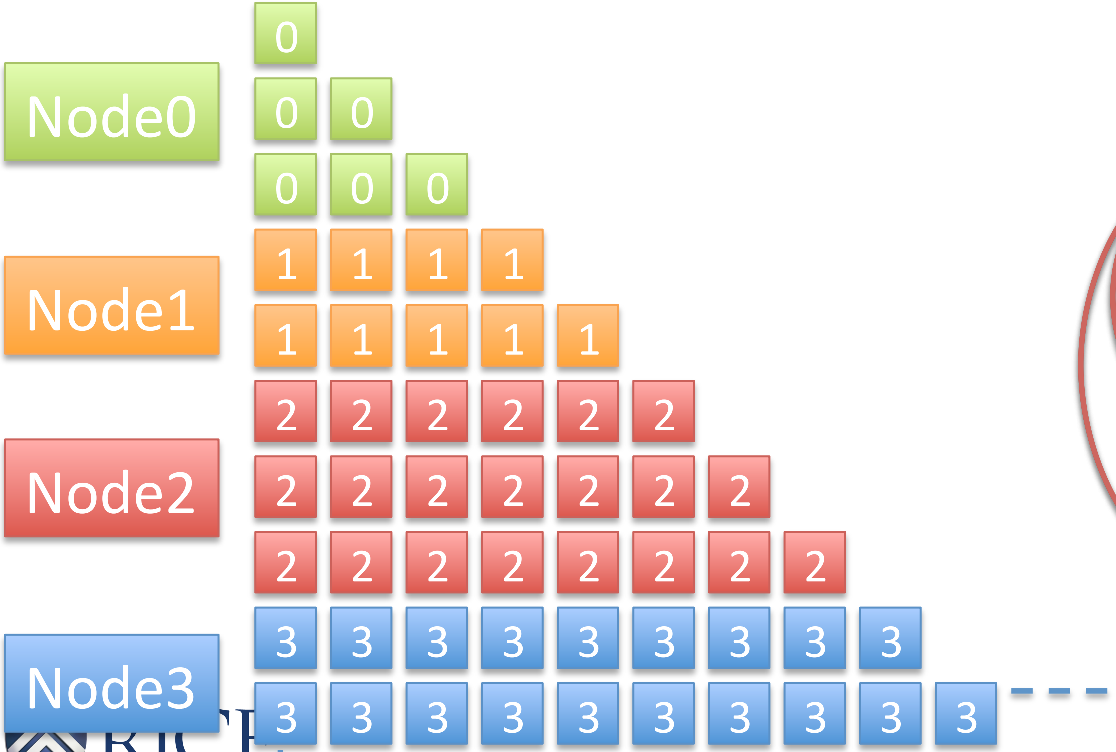


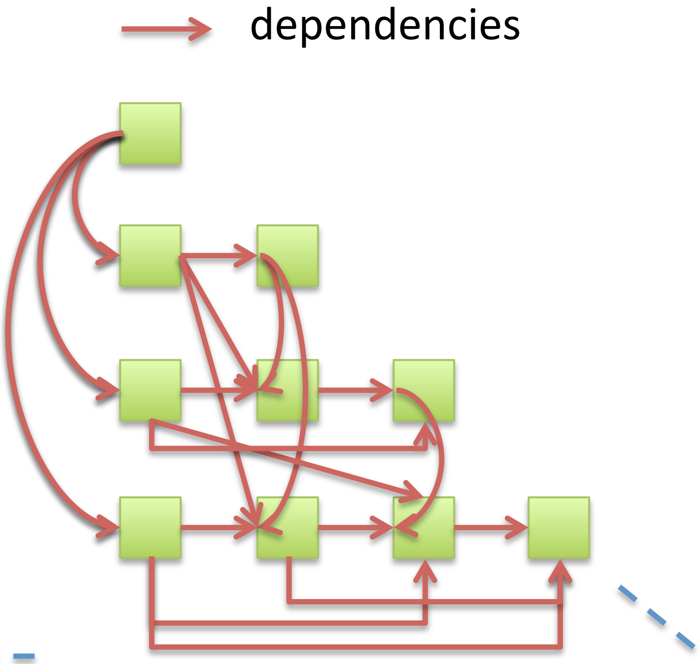
✓ *4.4x performance improvement relative to LLVM-unopt on the same # of locales on average (1, 2, 4, 8, 16, 32, 64 locales)*

Performance Evaluation

DETAILED RESULTS & ANALYSIS OF CHOLESKY DECOMPOSITION

Cholesky Decomposition

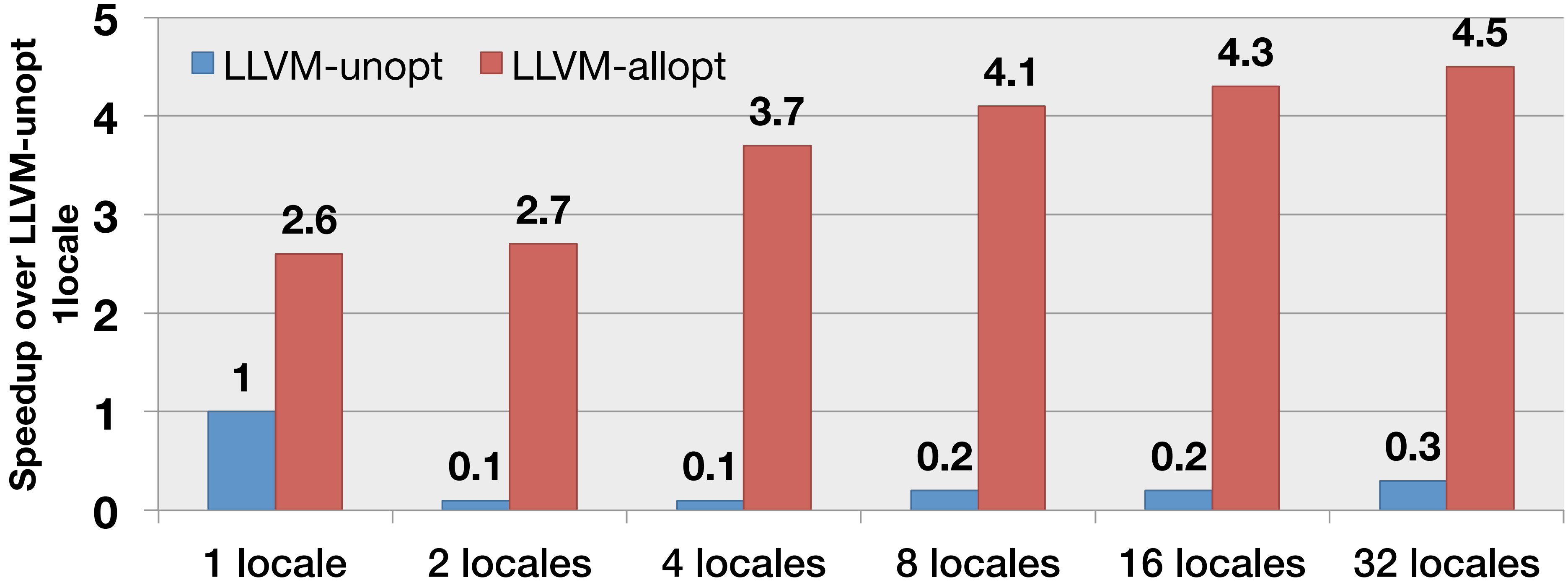




Metrics

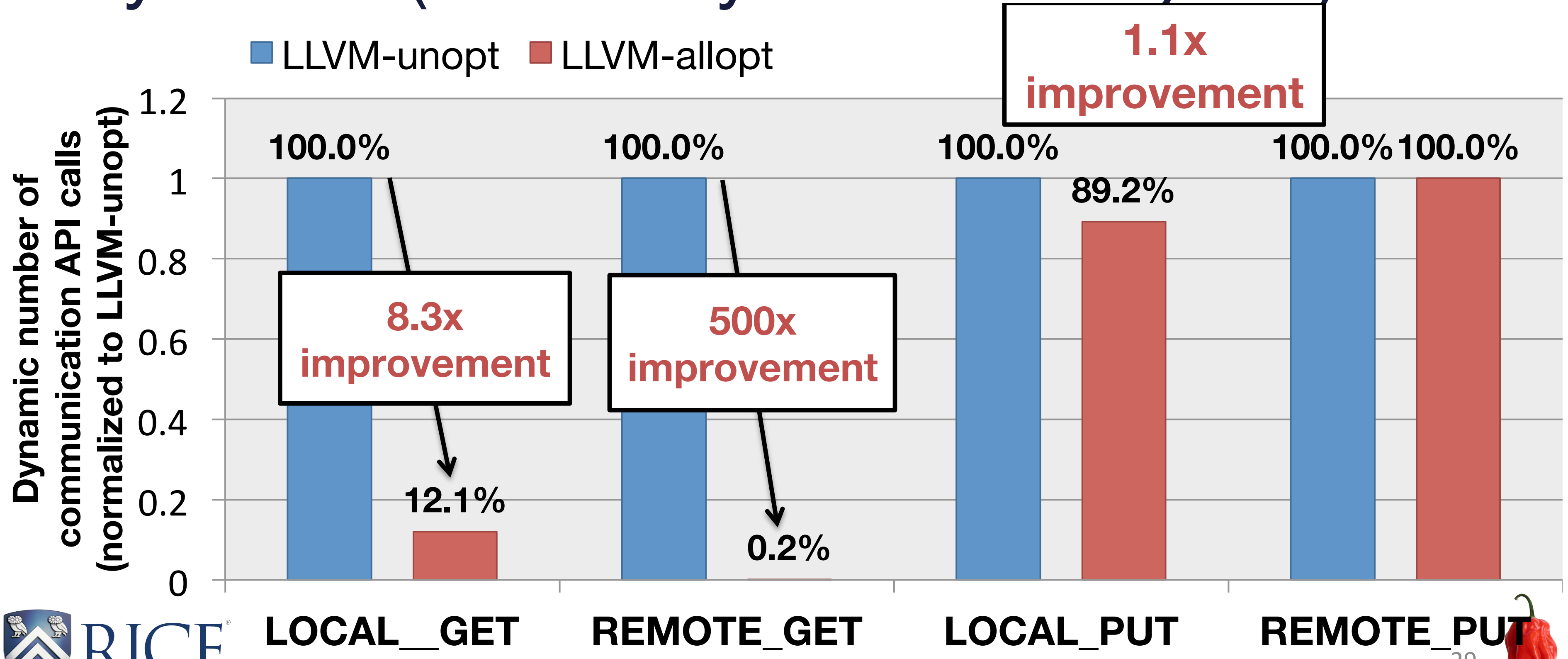
1. Performance & Scalability
   - Baseline (**LLVM-unopt**)
   - LLVM-based Optimizations (**LLVM-allopt**)
2. The dynamic number of communication API calls
3. Analysis of optimized code
4. Performance comparison
   - Conventional C-backend vs. LLVM-backend

Performance Improvement by LLVM (Cholesky on the Cray XC)



✓ ***LLVM-based communication optimizations show scalability***

Communication API calls elimination by LLVM (Cholesky on the Cray XC)



Analysis of optimized code

LLVM-unopt

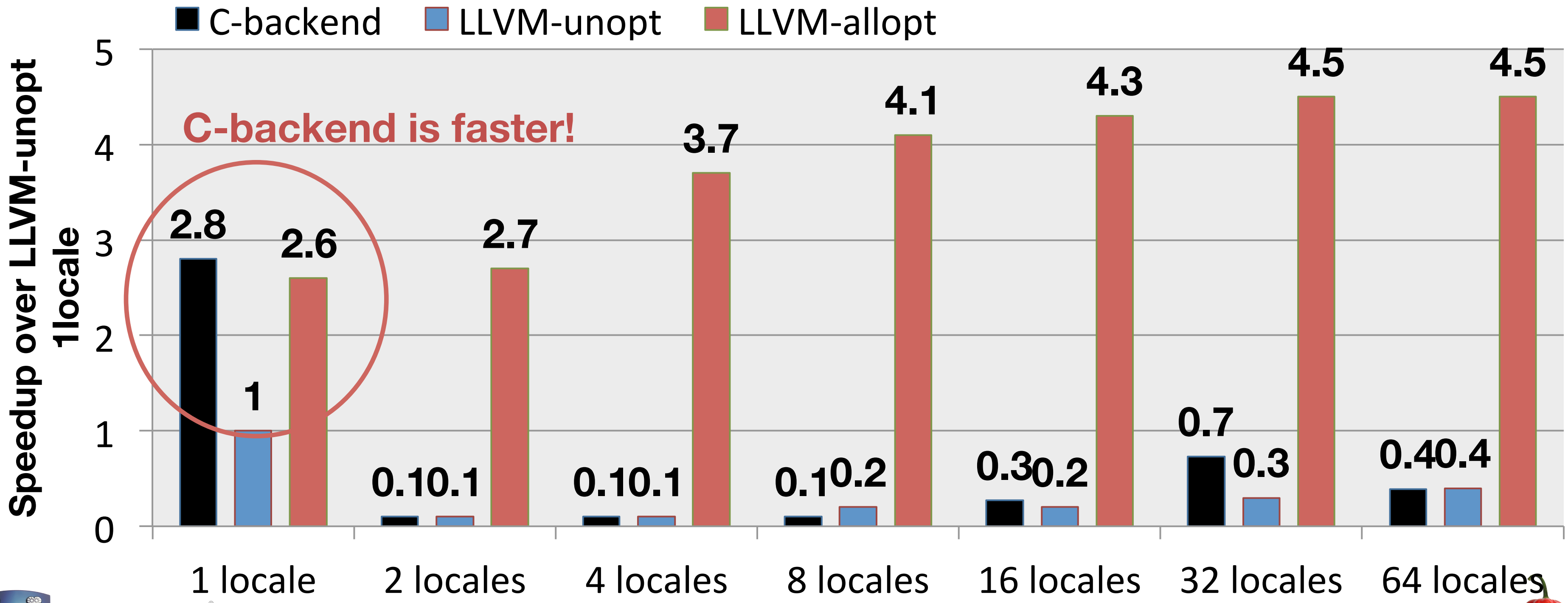
```
for jB in zero..tileSize-1 {
 for kB in zero..tileSize-1 {
  4GETS
  for iB in zero..tileSize-1 {
      9GETS + 1PUT
}}}
```

LLVM-allopt

1. **ALLOCATE LOCAL BUFFER**
2. **PERFORM BULK TRANSFER**

```
for jB in zero..tileSize-1 {
 for kB in zero..tileSize-1 {
  1GET
  for iB in zero..tileSize-1 {
      1GET + 1PUT
}}}
```

Performance comparison with C-backend



Current limitation

For C Code Generation :
128bit struct pointer

```
ptr.locale;
ptr.addr;
```

For LLVM Code Generation :
64bit packed pointer

Locale (16bit)	addr (48bit)

```
ptr >> 48
ptr | 48BITS_MASK;
```

1. **Needs more instructions**
2. **Lose opportunities for Alias analysis**

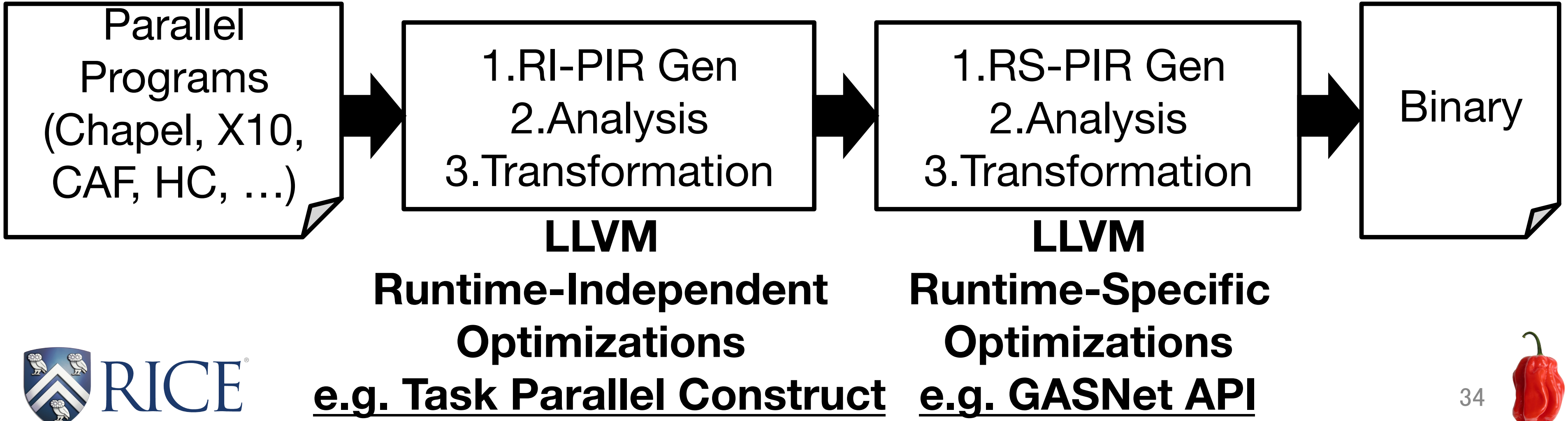
- In LLVM 3.3, many optimizations assume that the pointer size is the same across all address spaces

Conclusions

- LLVM-based Communication optimizations for PGAS Programs
  - Promising way to optimize PGAS programs in a language-agnostic manner
  - Preliminary Evaluation with 6 Chapel applications
    - ✓ Cray XC30 Supercomputer
      - 4.6x average performance improvement
    - ✓ Westmere Cluster
      - 4.4x average performance improvement

Future work

- Extend LLVM IR to support parallel programs with PGAS and explicit task parallelism
  - Higher-level IR



Acknowledgements

- Special thanks to
  - Brad Chamberlain (Cray)
  - Rafael Larrosa Jimenez (UMA)
  - Rafael Asenjo Plaza (UMA)
  - Habanero Group at Rice

Backup slides

Compilation Flow