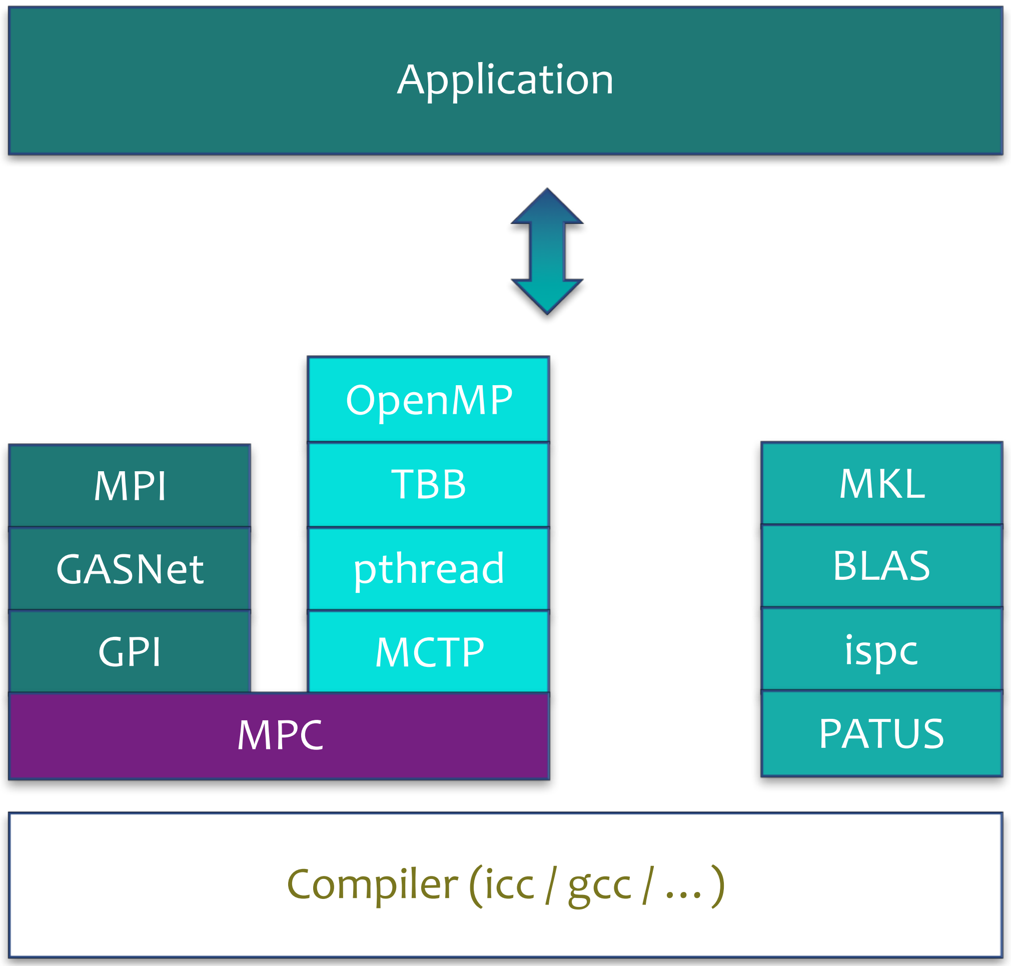
ExaShark

A modern high-level library for n-dimensional grids

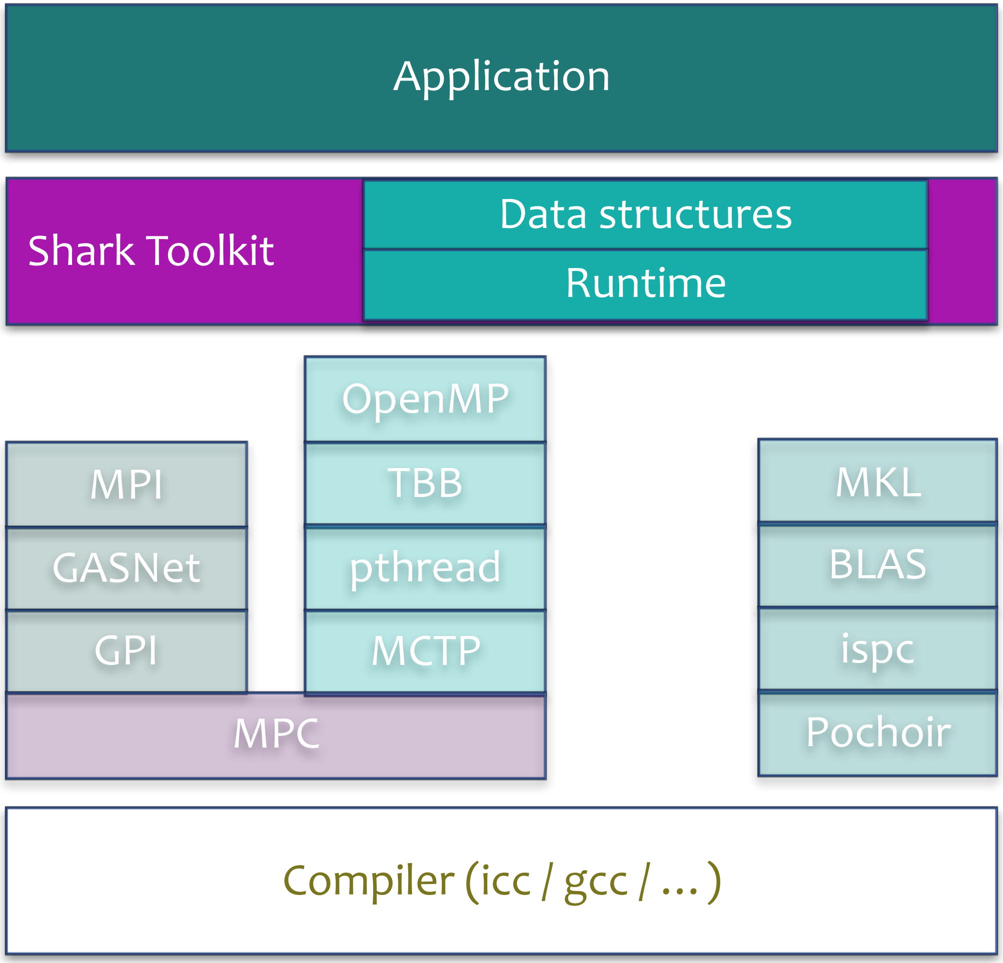
Tom vander Aa

Imen Chakroun, Pascal Costanza, Bruno Defraine, Tom Haber, Zubair Wadood, Roel Wuyts & The EXA2CT EU Project

Complicated Programming Stack



The Shark in the Middle



PGAS-style grids:
N-dimensional distributed grids
with local operations
Specific comm. patterns
Hybrid parallelism

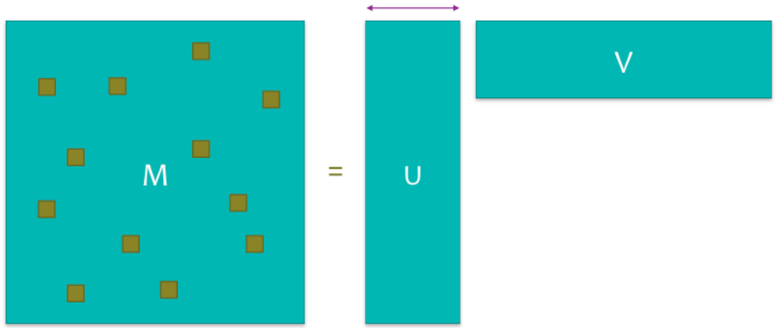
Inspired by:
Global Array (GA) Toolkit

Pacific Northwest NATIONAL LABORATORY

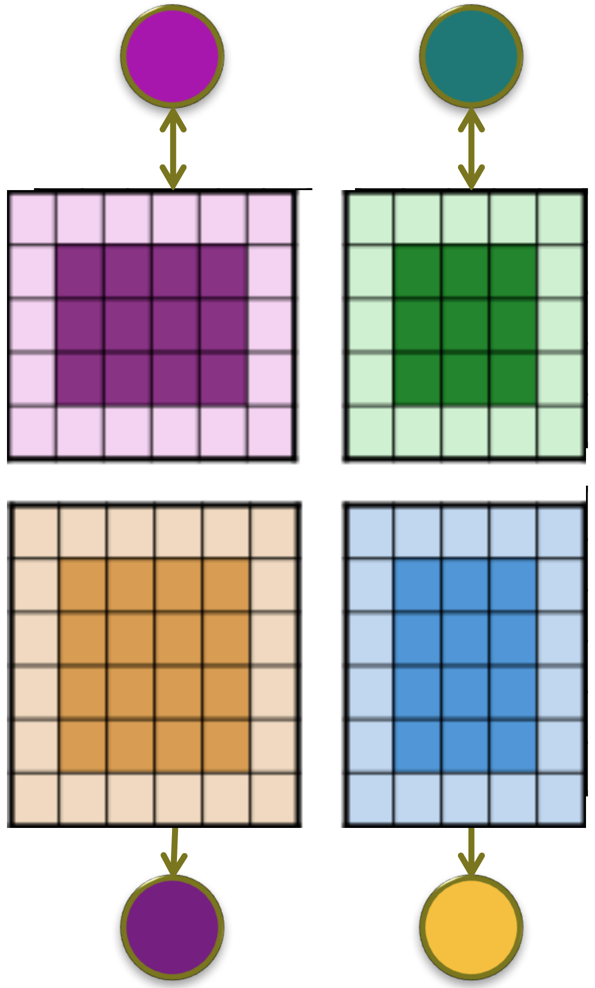
A bit about Shark

- Shark Technology
  - Shark Basics
  - C++11 features
  - Supported backends

- Applications built with Shark
  - Solvers
  - Benchmarks
  - HelSim PIC simulator
  - MACAU Recommender



GlobalArrays are Key

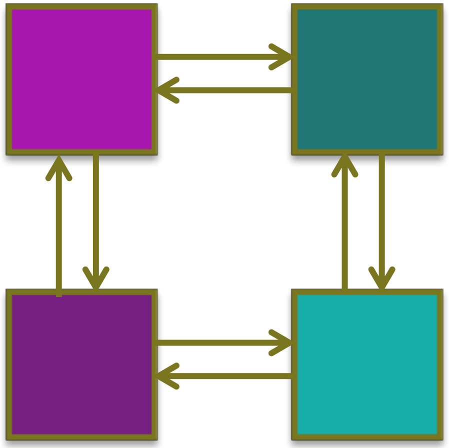


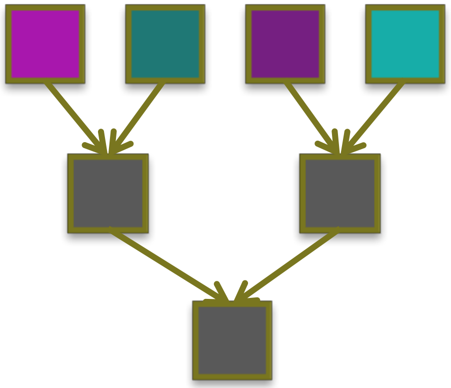
- GlobalArray:
  - Automatically or manually distributed
  - Ghost Borders
- Data-parallel Iterations
  - Locality Aware
  - C++ Expression Templates
    - **R = A + (B + C)**

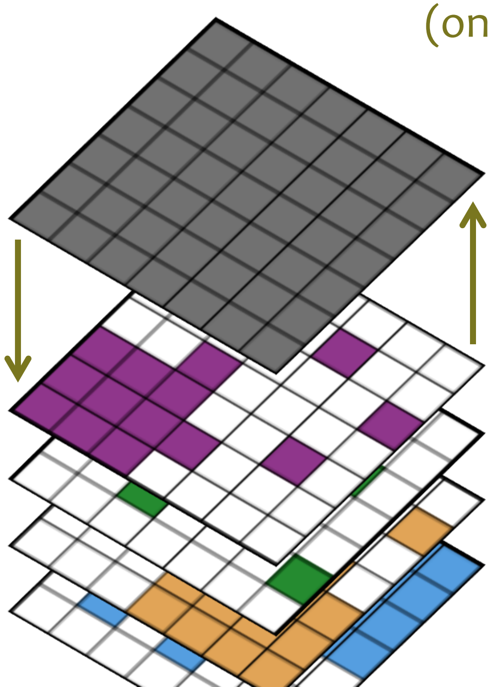
🟪	Matrix	●	Thread/process
▦	Ghost cells	↕	Memory access

Shark Communication Patterns

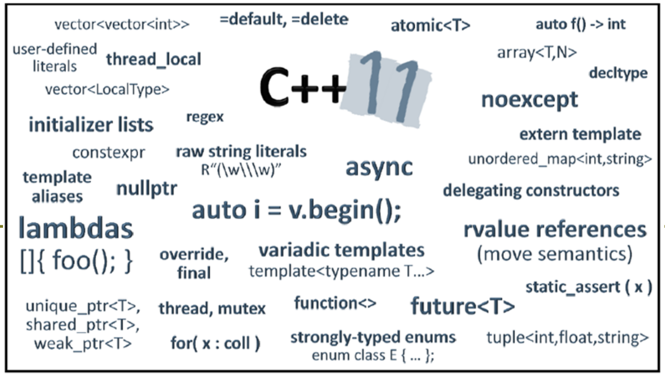
1. Ghost updates	geometric
2. Reductions	global
3. Gather/scatter with local array masks	long-distance (collective)
4. Get/put/accumulate RMA	long-distance (one-sided)







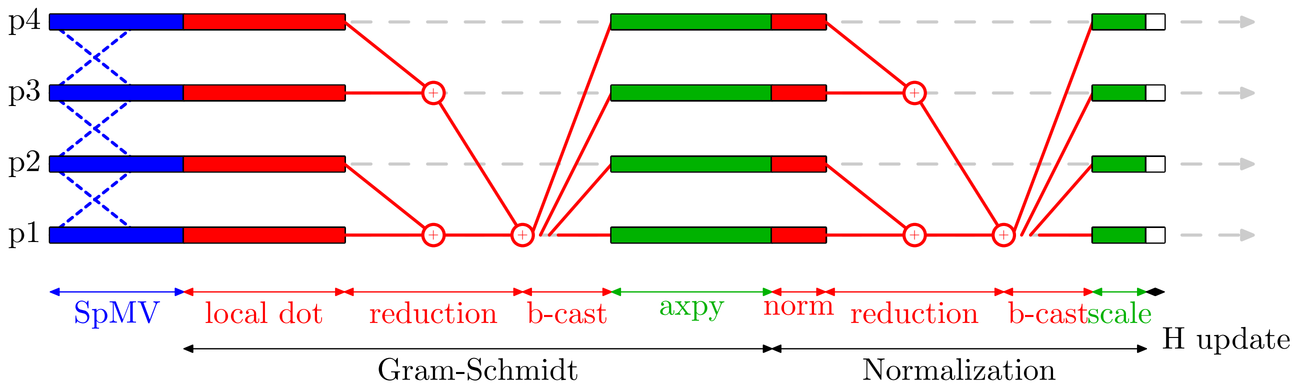
Built on C++11

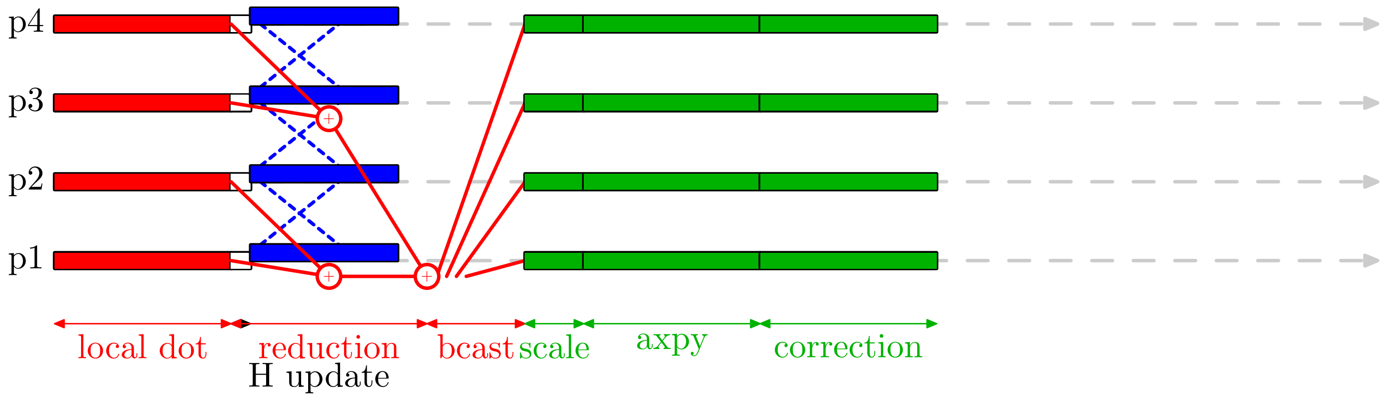


- Extensive template programming
  - Arbitrary number of dimensions
  - Any C++ type
  - Expression templates
    - Linear algebra ops in natural syntax
    - Implicitly data-parallel

- Non-blocking communication with Future<>

Shark for Solvers



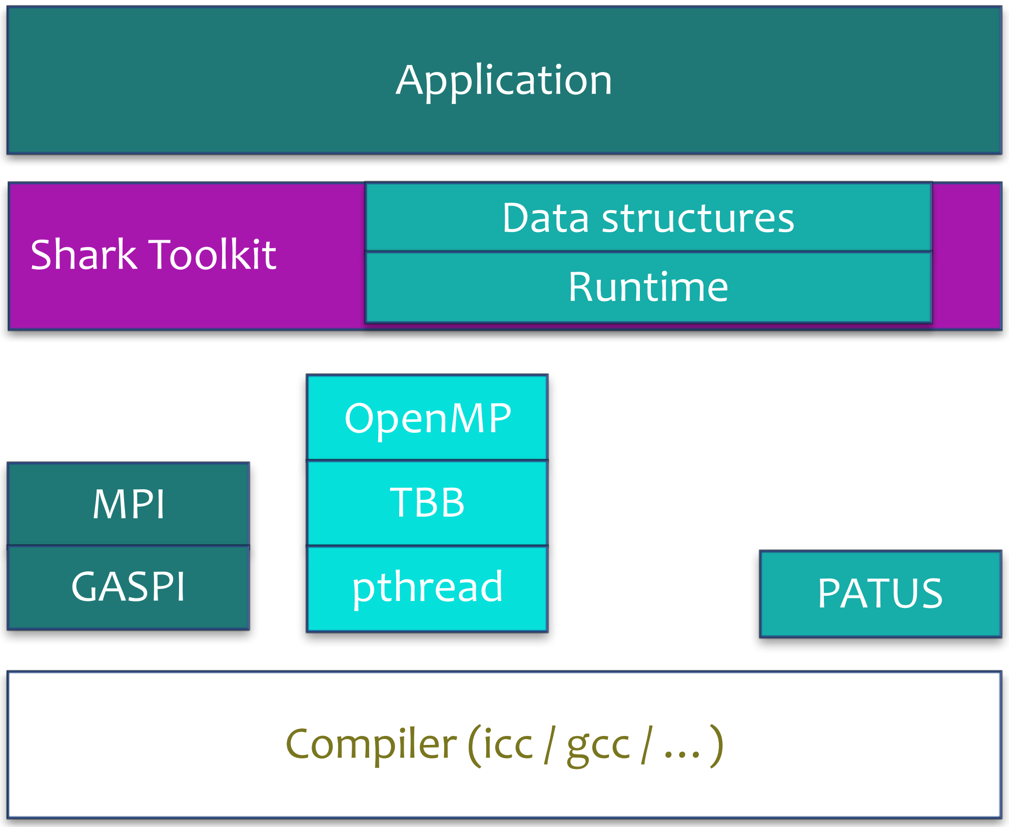


Nice Natural Syntax

```
1.  $r = b - Ax^{(0)}$
2.  $\rho_0 = ||r||_2$
3.  $k = 0,\ p = r,\ x = x^{(0)}$
4.  while $\rho \geq \epsilon$ and $k < k_{max}$
5.      $w = Ap$
6.      $\alpha = \rho_k^2/(p^T w)$
7.      $x = x + \alpha p$
8.      $r = r - \alpha w$
9.      $\rho_{k+1} = ||r||_2$
10.     $\beta = \rho_{k+1}^2/\rho_k^2$
11.     $p = r + \beta p$
12.     $k = k + 1$
```

```
r = b - Amult(x);
rho = norm2(r);
p = r;

for(k = 0; k < maxit; k++) {
    if(rho <= tol)
        break;

    w = A*p;

    alpha = rho*rho / dot(p,w);
    x = x + alpha * p;
    r = r - alpha * w;

    rho_old = rho;
    rho = norm2(r);

    beta = rho*rho / (rho_old*rho_old);
    p = r + beta * p;
}
```

Shark Supports Many Backends


Application
Shark Toolkit
Data structures
Runtime
OpenMP
MPI
TBB
GASPI
pthread
PATUS
Compiler (icc / gcc / …)