ExaSHARK+GASPI

Reducing the burden to program large HPC systems since 2014

Tom Vander Aa
Imen Chakroun, Mirko Rahn, Christian Simmendinger and Roel Wuyts &
The EXA2CT EU Project
EXA2CT
www.exa2ct.eu

10^{18}

Solvers that scale to ExaScale

Programming models that scale to ExaScale

Using relevant real-life prototype applications

TBB
CILK
PATUS
GPI
SHARK
Programming Models

- PATUS -- stencil compiler
- ExaSHARK -- n-dimensional grids
- GPI-2 – Asynchronous PGAS
Complicated Programming Stack

Application

MPI
GASNet
GPI
MPC
OpenMP
TBB
pthread
MCTP

MKL
BLAS
ispc
PATUS

Compiler (icc / gcc / ... )
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

Compiler (icc / gcc / ... )
A bit about Shark

• **Shark** Technology
  • Shark Basics
  • C++11 features
  • Supported backends

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

• The road to **ExaShark**
GlobalArrays are Key

- **GlobalArray:**
  - Automatically or manually distributed
  - Ghost Borders

- Data-parallel Iterations
  - Locality Aware
  - C++ Expression Templates

[Diagram showing matrices and thread/process relationships]
Shark Communication Patterns

1. Ghost updates
2. Reductions
3. Gather/scatter with local array masks
4. Get/put/accumulate RMA

- geometric
- global
- long-distance (collective)
- 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<>
Communication for SpMV can be overlapped with stencil calculations. For modified Gram-Schmidt, \((i+1)\) global reductions typically a binomial tree reduction with height \(d \log_2(P)\). Processor idling during all reduce latency has some additional flops compared to standard GMRES. Still waiting if reduction takes longer than SpMV.
1. \( r = b - Ax^{(0)} \)
2. \( \rho_0 = \|r\|_2 \)
3. \( k = 0, \; p = r, \; x = x^{(0)} \)
4. \textbf{while} \( \rho \geq \epsilon \) \textbf{and} \( k < k_{\text{max}} \)
5. \( w = Ap \)
6. \( \alpha = \rho_k^2 / (p^Tw) \)
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. \( k = k + 1 \)

\[
r = b - \text{Amult}(x);
\rho = \text{norm2}(r);
p = r;
\text{for}(k = 0; \; k < \text{maxit}; \; k++) \{ 
    \text{if}(\rho <= \text{tol}) 
    \text{break;}
\}
\]
\[
w = A*p;
\alpha = \rho*\rho / \text{dot}(p,w);
x = x + \alpha * p;
r = r - \alpha * w;
\rho_{\text{old}} = \rho;
\rho = \text{norm2}(r);
\beta = \rho*\rho / (\rho_{\text{old}}*\rho_{\text{old}});
p = r + \beta * p;
\]
Shark Supports Many Backends

Application

Shark Toolkit

Data structures

Runtime

OpenMP

TBB

pthread

MPI

GASPI

Compiler (icc / gcc / ...)

PATUS
GASPI in a nutshell

PGAS API - designed to be

- Multithreaded
- Heterogeneous architectures
- NUMA architectures
- Global asynchronous dataflow
- Interoperability with MPI
<table>
<thead>
<tr>
<th></th>
<th>ExaShark</th>
<th>GASPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trivial</td>
<td>GlobalArray()</td>
<td>gaspi_segment_create</td>
</tr>
<tr>
<td>Enlightening</td>
<td>Future&lt;&gt;</td>
<td>gaspi_notify/gaspi_notify_wait</td>
</tr>
<tr>
<td>Minimal Thinking</td>
<td>Put, Get</td>
<td>gaspi_write_block</td>
</tr>
<tr>
<td>PITA</td>
<td>Scatter, Gather, Accumulate</td>
<td>RequestQ&lt;AccContext&lt;GlobalArray&lt;ndim,T&gt;&gt;, AccRequest&lt;GlobalArray&lt;ndim,T&gt;&gt;&gt; q(AccContext&lt;GlobalArray&lt;ndim,T&gt;&gt;{src,acc});</td>
</tr>
<tr>
<td>Impossible</td>
<td>IDs</td>
<td>Voodoo Magic</td>
</tr>
</tbody>
</table>
SHARK+GASPI, the mess
From Shark to ExaShark

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

- The road to **ExaShark**
  - ExaScaling is Hard
  - Pure GASPI/MPI vs Shark
  - Scaling Shark
Strong Exa-Scaling is Hard

- CFD Application
  - Today: 50M mesh points
  - In ten years: 500M
- ExaScale
  - Many more cores, less memory per core
  - 50 mesh points per core
- CFD Proxy Application
  - Proto application of EXA2CT
CFD-Proxy on Xeon-Phi

![Graph showing speedup vs cores for various MPI operations.](image-url)
Strong Exa-Scaling is Possible

**Don’ts**
- Bulk Synchronous
- Single Threaded Communication
- MPI Data Types

**Dos**
- Completely Asynchronous
- GASPI write+notify
- MPI ISend/IRecv
- Thread-to-thread communication
- Multi-threaded packing
SHARK on par with pure MPI

3D Poisson CG, 400³ grid, 4x467 iterations

performance (iterations per sec)

cores

MPI
Shark
Shark MT
Weak scaling on stencils
ExaScaling IS hard

The graph shows the speedup for different configurations of nodes and cores. The configurations include:

- **nocomm**
- **gpi**
- **mpi+openmp**
- **mpi+ppn**

The x-axis represents the number of nodes and cores, while the y-axis represents the speedup. The graph highlights the performance differences among these configurations.
Conclusion on SHARK

• Modern high-level library for n-dim. grids
  • Support for many programming paradigms
  • Natural syntax thanks to C++
  • Performance good at node-level

• Much room for improvement
  • Active development
  • Support for GASPI/MPI Hybrid
  • Improve Scaling
    • More fine-grained a-synchronization
    • Multi-threaded Communication
Solvers that scale to ExaScale

Programming models that scale to ExaScale

Using relevant real-life proto applications

10^{18}