Redwood – Towards clusterless supercomputing in the cloud

Presented by Philipp A. Witte
Microsoft Research for Industry (RFI)
Microsoft Research for Industry (RFI)

- Bridge the gap between Industry R&D and Microsoft R&D
- Bring together MSR + Academia
- Unsolved problems, Tackle on Azure
- Bring innovations to Microsoft Products
- Contribute to Industry Innovations
Microsoft Research for Industry (RFI)

Microsoft RFI Team

- Ranveer Chandra
  Managing Director, Research for Industry Partner, Machine Learning Research, CTO, Agile Road
- Anirudh Badam
  Principal Research Scientist
- Tushar Chakraborty
  Software Engineer II
- Renato Luiz de Freitas Cunha
  Senior Research Software Development Engineer
- Pooijith Kumar
  Senior Research Scientist
- Ram Nagarjuna
  Principal Program Manager
- Peder Olsen
  Principal Researcher
- Riyaz Pishori
  Principal Program Manager
- Nikunj Raghuvanshi
  Senior Principal Researcher
- Upena Singh
  Principal Architect
- Philipp Witte
  Researcher
- Qie Zhang
  Principal Data & Applied Scientist

Academic Collaborations (CCS)

- Professor Sally Benson
  Stanford University
  CO2 modeling
- Professor Felix J. Herrmann
  Georgia Tech
  Seismic inversion, Devito & AI
- Professor Gerard J. Gorman
  Imperial College London
  Devito, HPC

RFI CCS Interns 2021

- Harpreet Kaur
  UT Austin
  Seismic inversion & AI
- Tugrul Konuk
  Colorado School of Mines
  Seismic inversion & AI
Challenges of HPC in the cloud

- Users need to manage HPC infrastructure
- Scalable & resilient HPC: only as strong as weakest link
Vision for HPC in the cloud

• Serverless computing

  + No infra management
  + Focus on application
  + Fast development
  + Usage-based billing

- Very limited hardware
- No orchestration
- Too limited in scope

• Clusterless HPC

  + All features of serverless
  + Run on any hardware
  + Orchestration + resilience
  + Not just an extension of serverless
  + Enable wide adoption of HPC
Redwood: Towards clusterless HPC

Conventional HPC

```
@everywhere function hello_world()
    print("Hello world")
end
@spawn hello_world()
```

Clusterless HPC with Redwood

```
@batchdef function hello_world()
    print("Hello world")
end
@batchexec hello_world()
```

HPC Cluster

- Head node
- Master
- Worker 1
- Worker 2
- Worker 3

TCP/IP

Azure Batch

- Master
- Worker 1
- Worker 2
- Worker 3

REST
Redwood: Towards clusterless HPC

@batchdef function hello_world()
    print("Hello world")
end
@batchexec hello_world()

{  
    "POOL_COUNT": "2",
    "NODE_COUNT_PER_POOL": "3",
    "INTER_NODE_CONNECTION": "1",
    "POOL_VM_SIZE": "Standard_E32s_v3"
}
Redwood – How does it work?

Application

```csharp
using Redwood

# Create pool
startup_script = "pool_start.sh"
create_pool_and_resource_file(startup_script)

# Define function
@batchdef function hello_batch(my_id)
    print("Hello World from node $m_yid")
end

# Execute via Azure Batch
@batchexec pmap(my_id -> hello_world(my_id), 1:4)
```

Redwood
- Closure
- Code gen.
- API calls

Redwood leverages Azure Batch:
- Pool management
- Execute tagged functions
- I/O, fault tolerance

Parameter JSON

```json
{
    "_POOL_VM_SIZE": "Standard_E8sv3",
    "_POOL_COUNT": "3",
    "_NODE_COUNT_PER_POOL": "8",
    "_MPI_RUN": "0"
}
```
# Function definition
@everywhere function hello_world(name; kwargs...)
    print("Hello ", name)
    return "Goodbye"
end

# Remote execution
future = @spawn hello_world("world")

# Fetch output
output = fetch(future)

---

# Function definition
@batchdef function hello_world(name; kwargs...)
    print("Hello ", name)
    return "Goodbye"
end

# Remote execution
bctrl = @batchexec hello_world("world")

# Fetch output
output = fetch(bctrl)
Acceleration through abstractions

Batch/MapReduce jobs

```python
using Redwood
# Hello world
@batchdef function hello_world()
    print("Hello world")
end

# Execute tasks in parallel
bctrl = @batchexec hello_world()

# Collect and reduce
output = fetchreduce(bctrl; op=+)
```
Acceleration through abstractions

```
using Redwood
@batchdef using MPI

# Define MPI function
@batchdef function hello_world()
    MPI.Init()
    comm = MPI.COMM_WORLD
    println("Hello world, I am rank $(MPI.Comm_rank(comm))\n")
    MPI.Barrier(comm)
end

# Execute three MPI jobs in parallel
bctrl = @batchexec pmap(i -> hello_world(), 1:3)
```
Acceleration through abstractions

```
using Redwood
@batchdef using Distributed

# Hello world
@batchdef hello_world() = print("Hello world")

# Define pmap function
@batchdef function say_hello()
    output = pmap(i -> hello_world(), 1:4)
    return output
end

# Execute three multi-task jobs in parallel
bctrl = @batchexec pmap(i -> say_hello(), 1:3)
```
Redwood – Examples & applications
(1) Seismic imaging (RTM)

# Packages
@everywhere using Distributed, Jets, JetPackWaveFD

# Functions
@everywhere function modelshot(isrc, sx, _v; kwargs...)
  ...
end

# Remote execution
shot = epmap(isrc -> modelshot(isrc, sx, _v; kwargs...), 1:10)

# Packages
@batchdef using Distributed, Jets, JetPackWaveFD

# Functions
@batchdef function modelshot(isrc, sx, _v; kwargs...)
  ...
end

# Remote execution
bctrl = @batchexec pmap(isrc -> modelshot(isrc, sx, _v; kwargs...), 1:10)
shot = fetch(bctrl)
(1) Seismic imaging (RTM)

Redwood scalability
- Enable multiple batch pools and/or accounts
- Scheduling of jobs across many pools
- RTM using $16 \times 256 = 4,096$ VMs
(2) Scale across continents

```
In [2]: # Create pool
startup_script = "pool_startup_script_cofii.sh"
create_pool_and_resource_file(startup_script);

Created pool 1 of 10 in australiacentral with 2 nodes.
Created pool 2 of 10 in brazilsouth with 2 nodes.
Created pool 3 of 10 in eastasia with 2 nodes.
Created pool 4 of 10 in eastus2 with 2 nodes.
Created pool 5 of 10 in francecentral with 2 nodes.
Created pool 6 of 10 in koreasouth with 2 nodes.
Created pool 7 of 10 in southafricannorth with 2 nodes.
Created pool 8 of 10 in southeastasia with 2 nodes.
Created pool 9 of 10 in uae with 2 nodes.
Created pool 10 of 10 in westus2 with 2 nodes.
```

```
In [3]: @elapsed begin
result = optimize(Optim.only_fg!(_fg!), v2, solver, Optim.Options(
    iterations = niter,
    callback = mycallback
)
end
```

Quasi Newton optimization algorithm from Optim.jl

“Global FWI”
- Globally distributed pools
- Run FWI on 6 continents
- Harvest resources across globe
### (3) Sound simulation

- Project Triton from MSR
- Sound simulations for games
- Simulate **14,197** probes in parallel

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Desktop</th>
<th>Single batch pool</th>
<th>Redwood</th>
</tr>
</thead>
<tbody>
<tr>
<td>14,197</td>
<td>147 days</td>
<td>3.5 hours</td>
<td>0.25 hours</td>
</tr>
</tbody>
</table>

15,000 node cluster
(4) Reservoir simulation

- **Input X**: Permeability
- **Output Y**: Saturation history

1,000 training pairs

Network prediction (SNR 12.86)
AzureClusterlessHPC.jl - Simplified distributed computing

Overview

AzureClusterlessHPC.jl is a package for simplified parallel computing on Azure. AzureClusterlessHPC.jl borrows the syntax of Julia's Distributed Programming package to easily execute parallel Julia workloads in the cloud using Azure Batch. Instead of a parallel Julia session, users create one or multiple worker pools and remotely execute code on them.

https://github.com/microsoft/AzureClusterlessHPC.jl