Санкт-Петербургский
государственный
университет

Distributed Virtual Cluster Management System

Amissi Cubahiro, Vladimir Gaiduchok, Magdalyne Kamande,
Sergey Kobyshev, Vladimir Korkhov, Alexander Degtyarev

St. Petersburg State University
v.korkhov@spbu.ru

GRID’18, Dubna, Russia, September 11, 2018

Motivation

- Make distributed computing system easier to **use** and to **manage**
- Allocate **as much** resources **as needed** by applications
- Enable **controlled** concurrent use of **shared resources** with minimal impact on application performance

Goal: provide user applications with access to as much resources as needed, and try to optimize shared resource usage

Applications

- Parallel applications in cloud-based distributed systems
  - e.g. MPI applications
- Frameworks for distributed data processing
  - e.g. Apache Hadoop

Approach

- Build **tailored virtual computing environments**:
  - Tune the computing infrastructure to optimize application performance and optimally distribute virtualized physical resources between applications – **application-centric** approach

- **Virtualization** of resources
  - Create **virtual clusters** that match application profiles (configurable CPU, memory, network)
  - Use **light-weight virtualization** with less overhead
  - Enable **flexible** configuration of infrastructure

- Different **applications** have different **profiles** and **requirements**

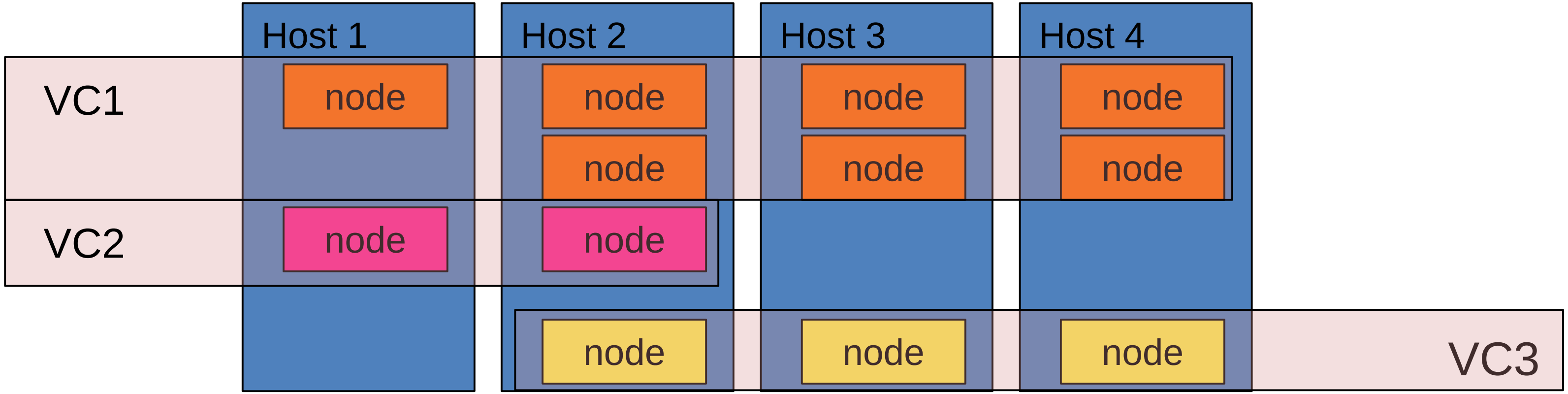
Base layer of the infrastructure

- hardware node;
- virtual machine;
- **container**

(a) Container-based Virtualization (b) Full or Para-Virtualization

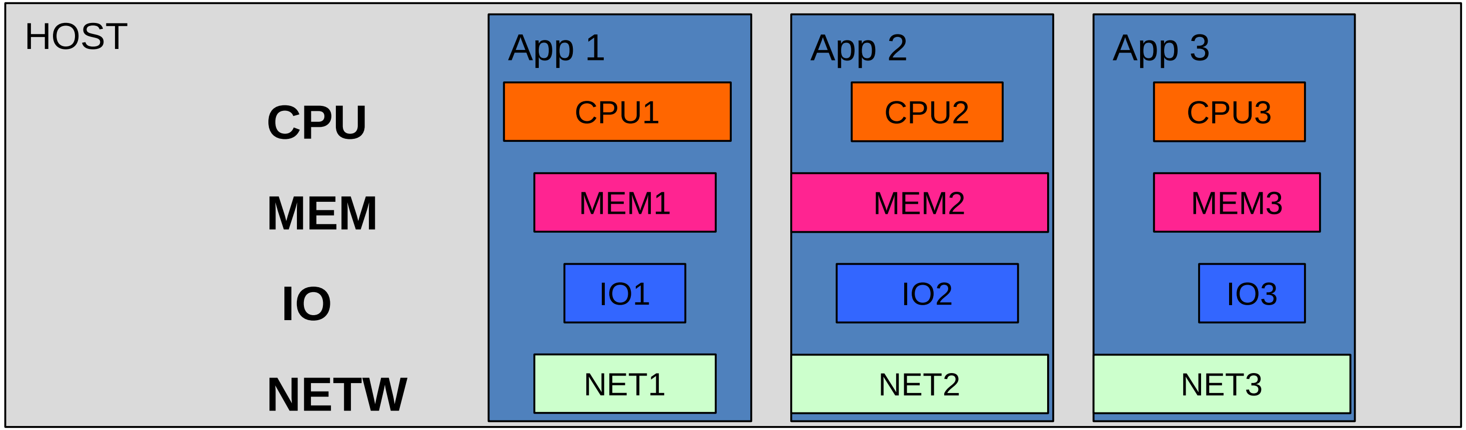
Virtual clusters

- Collection of virtual nodes working together to solve a computational problem
- Can be configured by advanced users; they know exactly what they want (CPU, memory, IO, network)
- Can be flexibly adjusted to the needs of an application

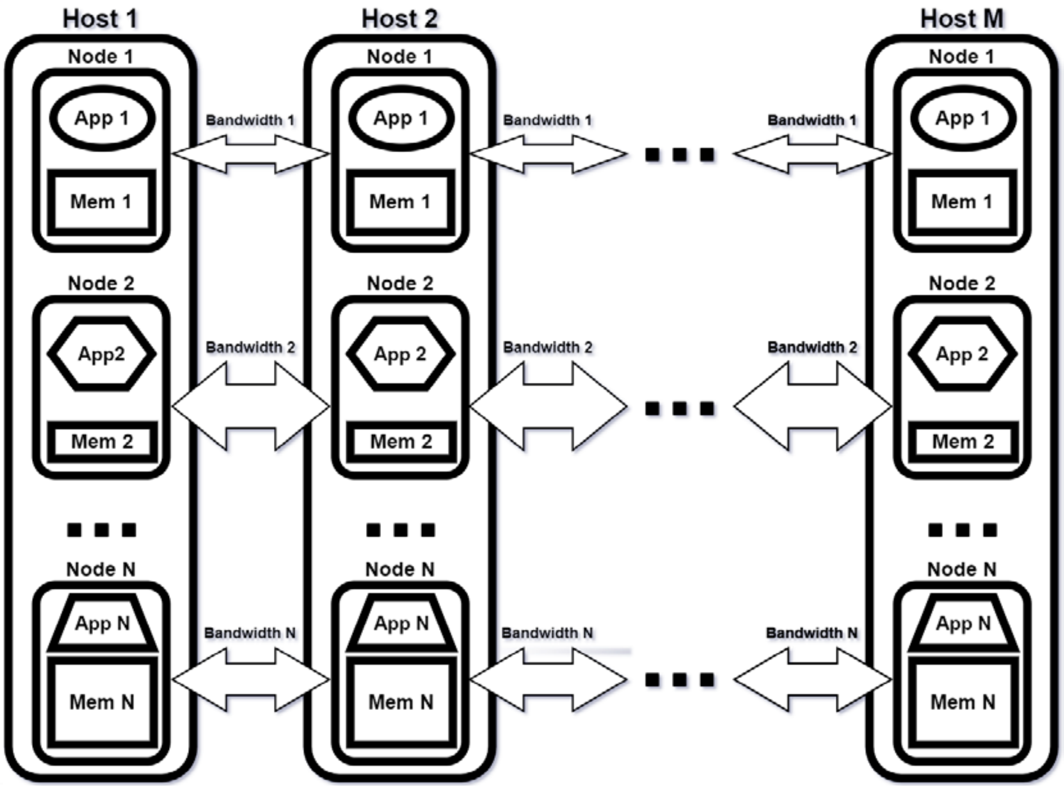


Why virtual clusters?

- Precise control on allocated resources (CPU, memory, etc)
- Applications get exactly what they need (or what they request): one app needs fast disk IO and not much CPU, another one – fast network and fast CPU with no disk IO, etc
- Capacity of unclaimed resources available for other applications on a limited set of hardware



Concurrent execution



Experiments with MPI applications

NAS Parallel Benchmarks (NPB)

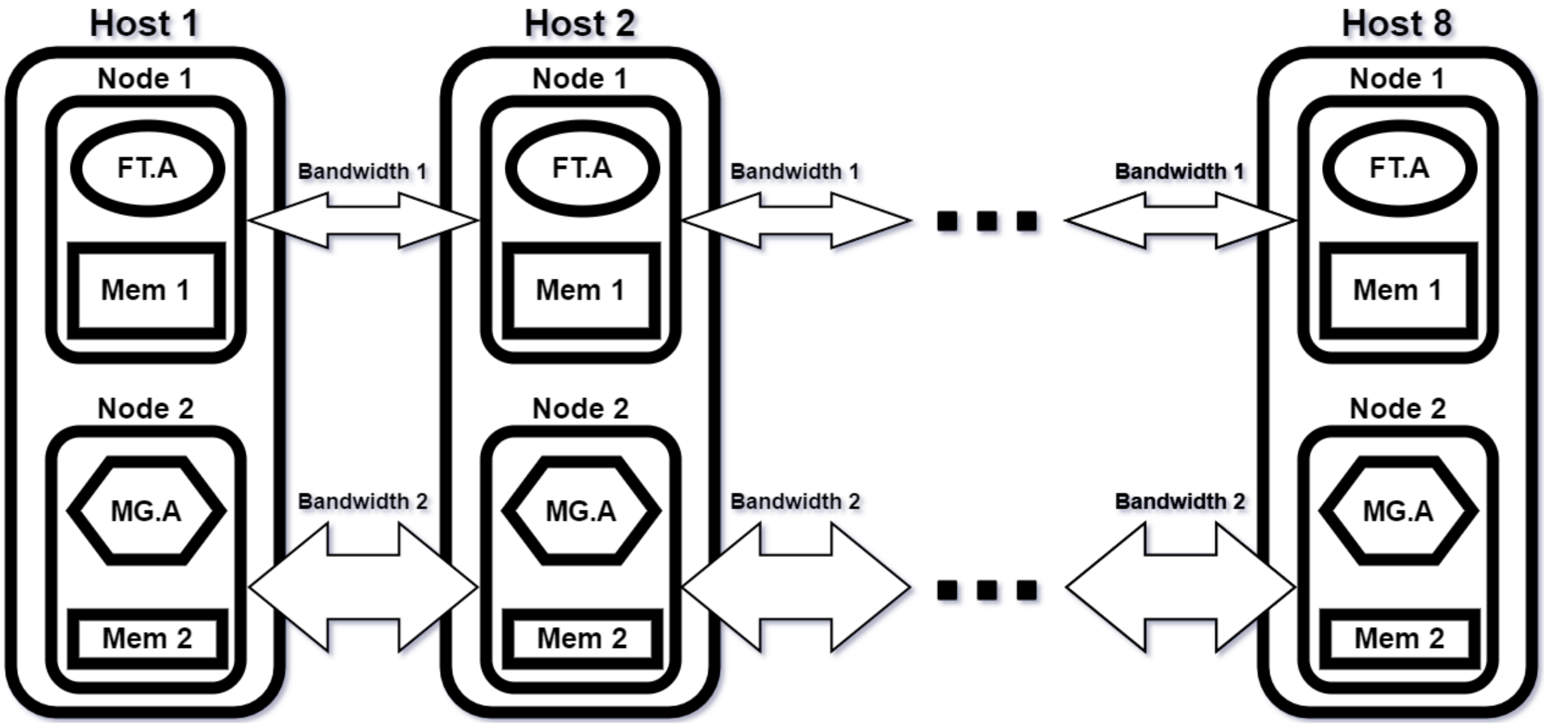
https://www.nas.nasa.gov/publications/npb.html

The benchmarks are derived from computational fluid dynamics (CFD) applications and consist of several kernels and pseudo-applications

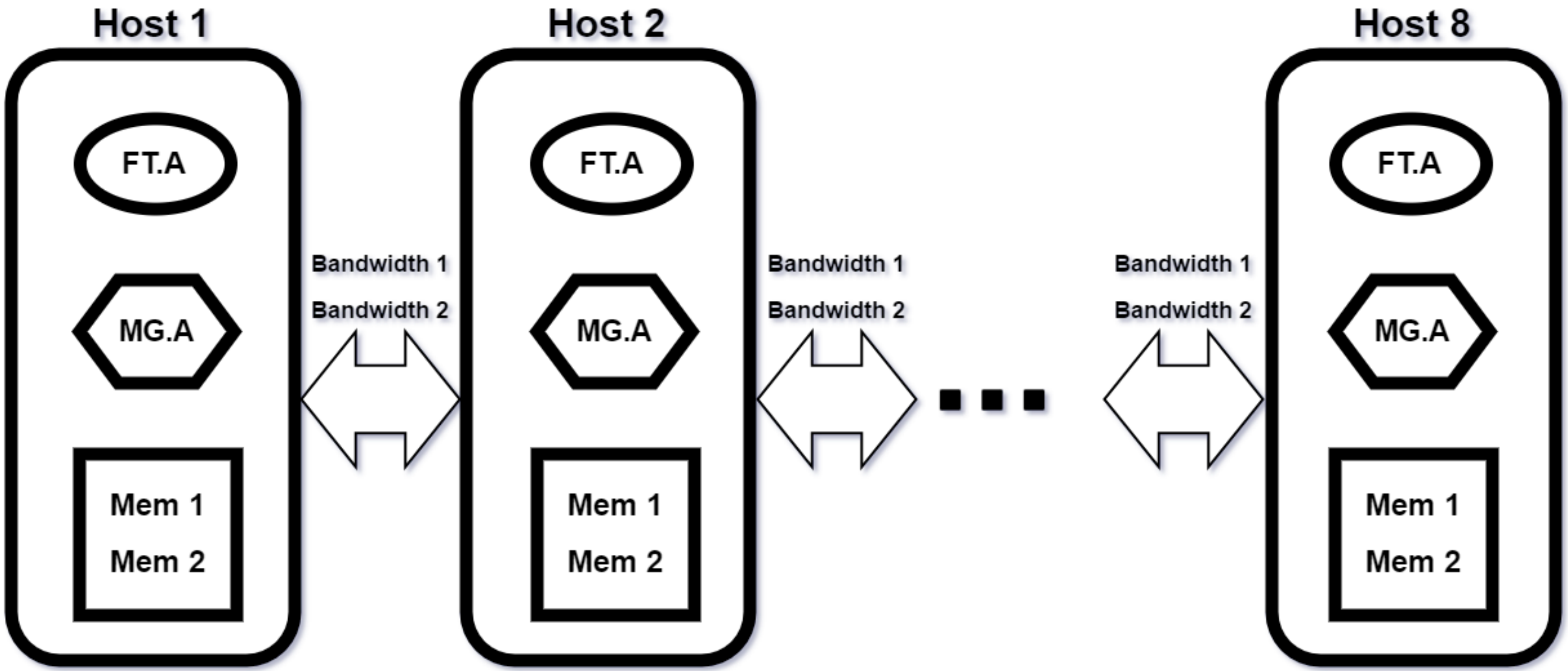
FT - discrete 3D fast Fourier Transform, all-to-all communication
CG - Conjugate Gradient, irregular memory access and communication
MG - Multi-Grid on a sequence of meshes, long- and short-distance communication, memory intensive

Experimental testbed: 8 nodes of MS Azure resources
Docker Swarm for managing container clusters

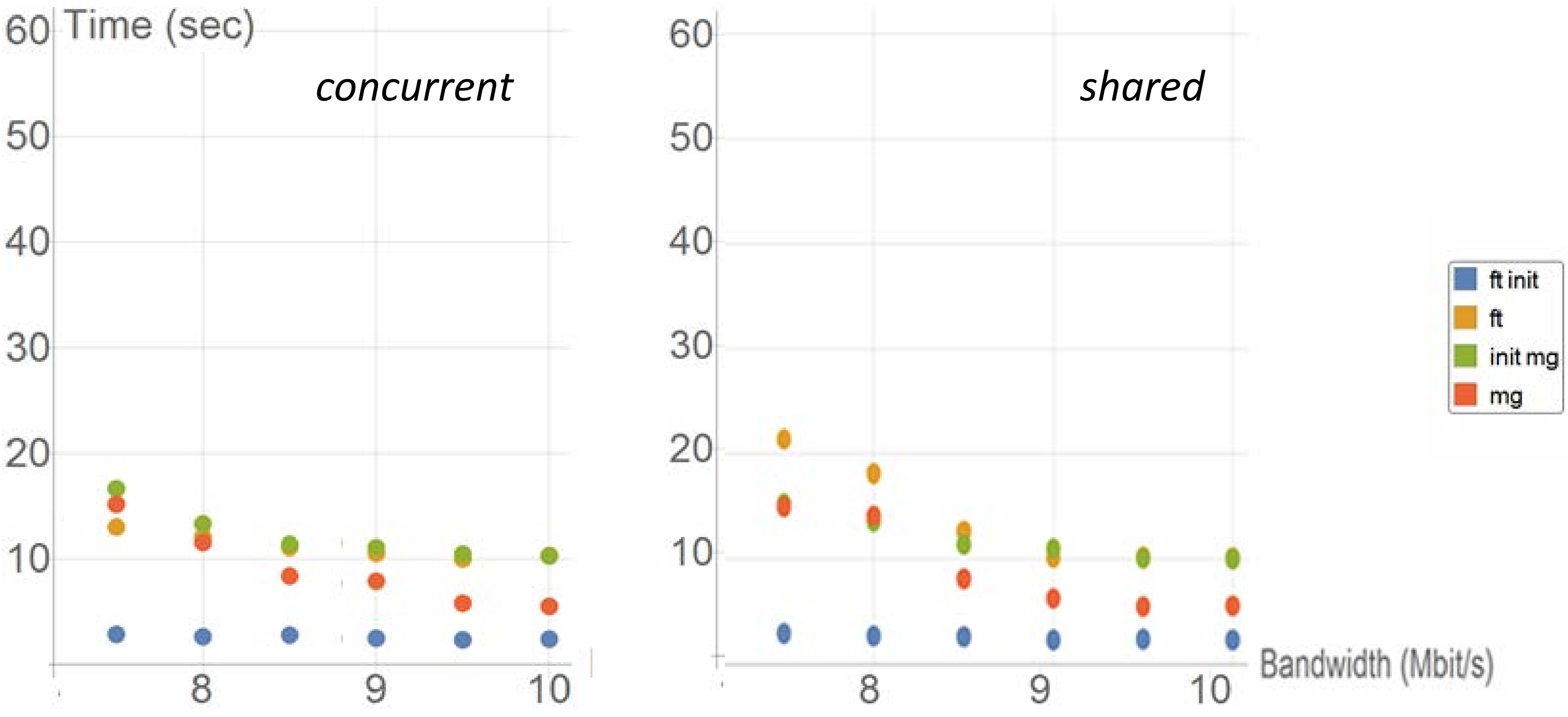
Work pattern (FT.A + MG.A), concurrent



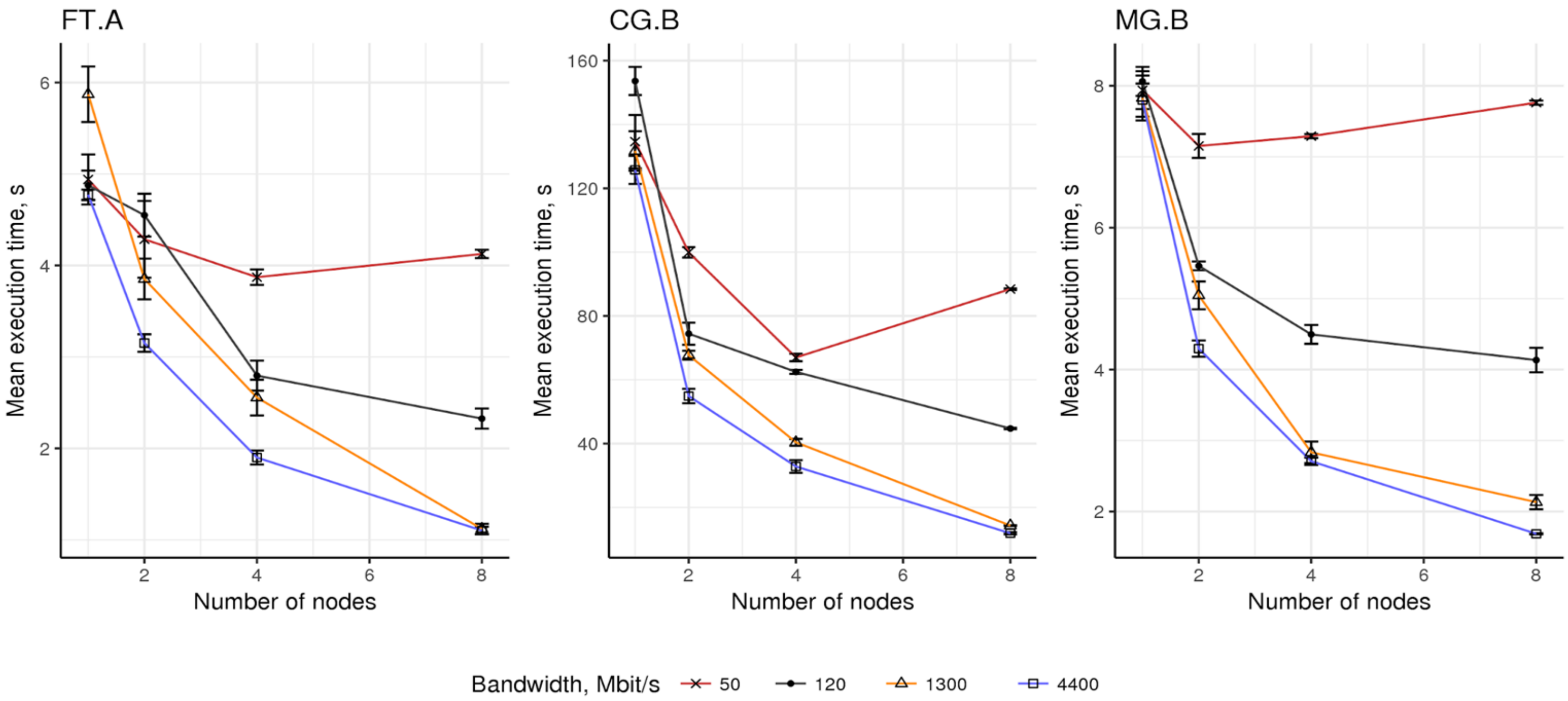
Work pattern (FT.A + MG.A), shared



Results (FT.A + MG.A), concurrent vs shared

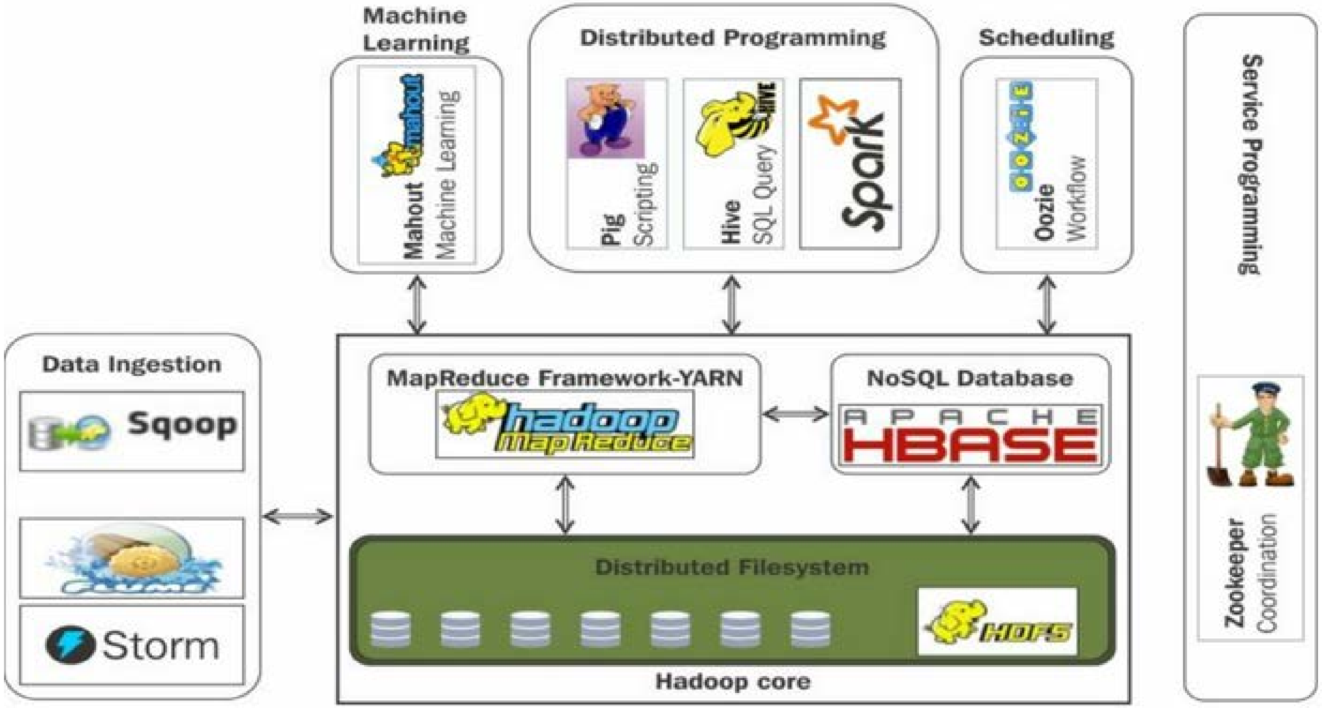


Optimal resource configuration



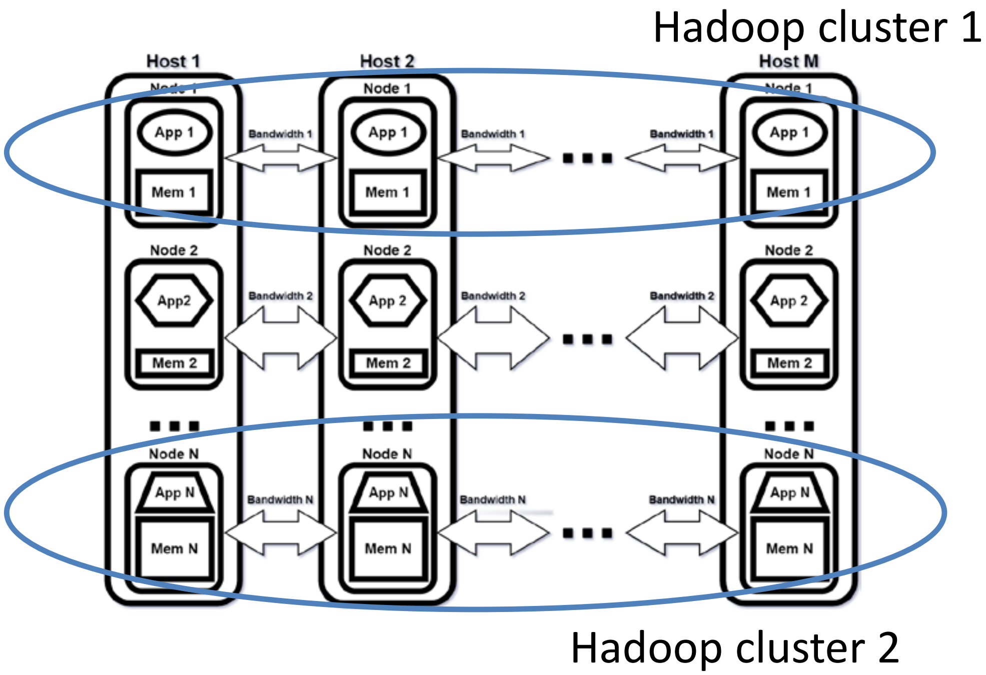
More details: *this Thursday, in* R. Kuchumov, V. Korkhov. Design and implementation of a service for performing HPC computations in cloud environment

Experiments with Hadoop



Hadoop on virtual clusters

Amazon AWS VMs: t2.large and t2.medium instances
Docker Swarm: one or more containers on VM
CloudPly: creating light-weight virtual infrastructures on top of physical or cloud resources - configuration of nodes, app deployment



Benchmarking Hadoop on VC: test suites

TestDFSIO

- read and write storage throughput test for HDFS

TeraSort

- performs significant computation, networking, and storage I/O workloads;
- combines testing the HDFS and MapReduce layers of a Hadoop cluster;
- often considered to be representative of real Hadoop workloads;
- divided into three parts: generation, sorting, and validation.

MRBench

- runs small jobs a number of times and checks whether small jobs are responsive

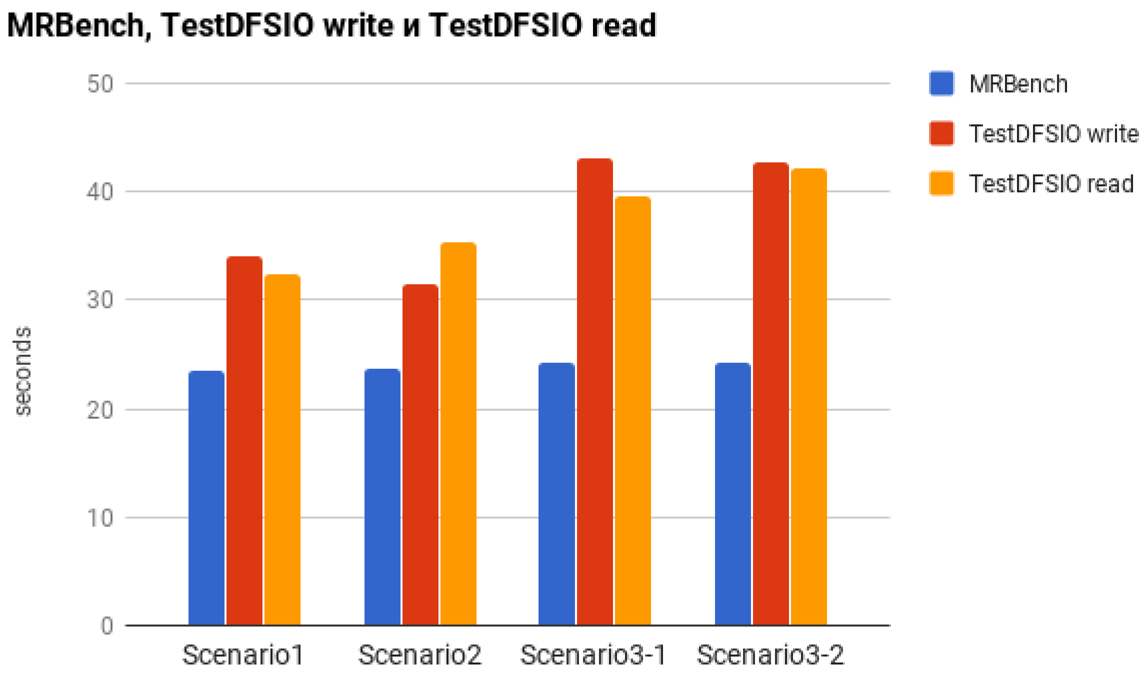
Benchmarking Hadoop on VC: scenarios

Scenario1: The cluster is composed a set of **t2.large VMs** (**2 vCPUs, 8GB RAM**); every VM runs a **single Docker container** that uses **full VM** resources without constraints; Hadoop is deployed with 1 namenode and 2 worker nodes;

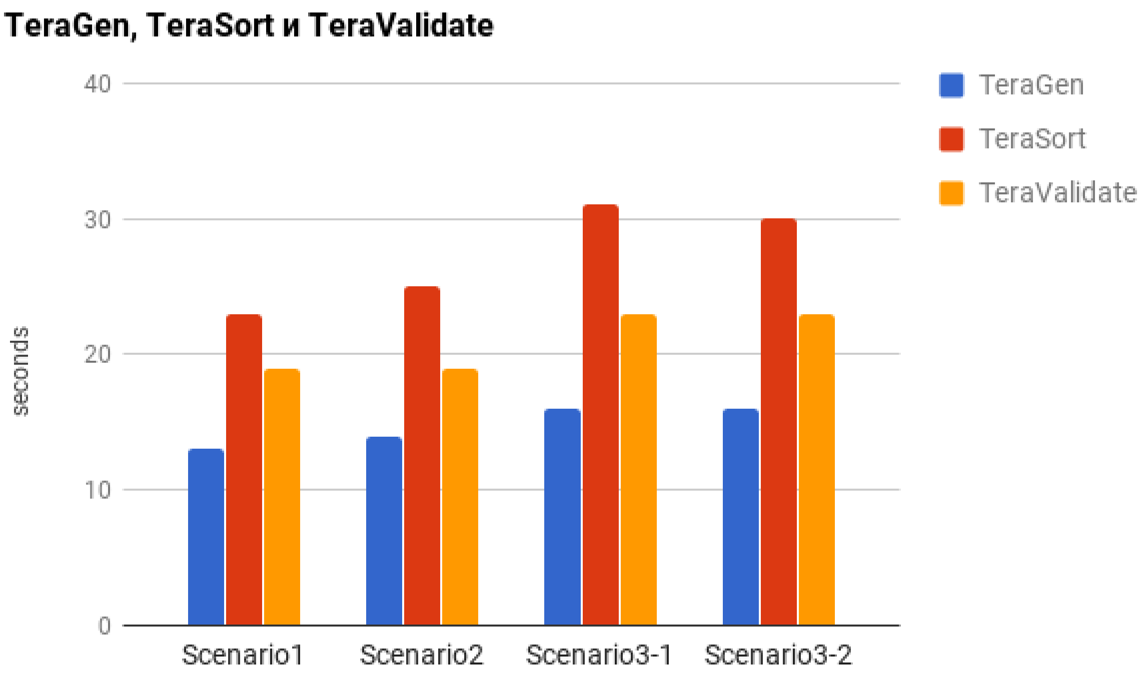
Scenario2: The cluster is composed a set of **t2.large VMs**; every VM runs a **single Docker container** constrained to use **only 4GB RAM**; Hadoop is deployed with 1 namenode and 2 worker nodes;

Scenario3: The cluster is composed a set of **t2.large VMs**; every VM runs **two Docker containers**, each constrained to use **only 4GB RAM**; **two Hadoop clusters** are deployed in parallel on containers 1 namenode and 2 worker nodes; thus **every VM is shared** between two simultaneously running Hadoop clusters.

Evaluation


MRBench, TestDFSIO write и TestDFSIO read
seconds
50
40
30
20
10
0
Scenario1
Scenario2
Scenario3-1
Scenario3-2
MRBench
TestDFSIO write
TestDFSIO read

Evaluation


TeraGen, TeraSort и TeraValidate
seconds
40
30
20
10
0
Scenario1
Scenario2
Scenario3-1
Scenario3-2
TeraGen
TeraSort
TeraValidate

Discussion

MRBench performance does not depend on the scenario: indeed, it focuses on MapReduce without much use of HDFS, thus it relies mostly on CPU. In our setup every VM has two vCPUs, thus even in scenario 3 each container gets its own CPU.

TestDFSIO significantly depends on the scenario: in Scenario 3 both read and write tests perform significantly slower than in Scenarios 1 and 2, though not twice as slow but only about 1.5 times slower, which supports the statement about efficiency of using parallel clusters.

TeraSort shows only a slight decrease of performance in Scenario 3: we managed to process twice as much as the original TeraSort workload increasing the overall processing time just for about 15 percent

Conclusion

- Deployed container clusters over Amazon VMs; performed experiments to check efficiency of using resources by *NAS Parallel Benchmarks* and *Hadoop benchmarks: MRBench, TestDFSIO, TeraSort*.
- Demonstrated that efficiency of using distributed resources can be increased - even in case of utilizing cloud resources - by *simultaneous execution of light-weight virtual clusters*
- Flexible configuration of container clusters with standard tools helps *allocate proper amount of resources* and control free available resources
- Need to profile (or model) applications to specify realistic requirements depending on input data (*more on this on Thursday*)
- Future work: look into more advanced tools for controlling containers to concentrate on application-specific infrastructure management

The work is supported by RFBR project 16-07-01111

Thank you for your attention!