Metrics & Benchmarks

Task 2.3

Marios Dikaiakos **George Tsouloupas**

Dept. of Computer Science

University of Cyprus

Nicosia, CYPRUS

Benchmarks 101

- Synthetic Codes and Application Kernels with well-defined and understood behaviour.
- Used for capturing performance properties of:
  - Hardware & architecture (NAS, SPEC, SPLASH…).
  - Operating System components.
  - Software systems and applications (e.g. TPC).
  - Various kinds of middleware (more recently).
- Testing conducted with simple (stress testing) or synthetic (mimicking reality) workloads.
- Testing used to capture performance properties (metrics) that characterize properties under observation, discover bottlenecks, etc.
- Properties are defined and interpreted in the context of models for hardware, architecture, available resources, etc.

CG Task 2.3: Mission

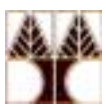
- Propose models for a Grid configuration, and metrics to capture and describe concisely:
  - The performance capacity of Grid configurations (VO's).
  - The performance of Grid applications.
- Propose a suite of *representative* benchmarks capturing these metrics.
- Implement and run benchmark-suite on CrossGrid test-bed.
- Make benchmarks and measurements publicly available and analyse results to assess the usefulness of benchmarks in:
  - Identifying factors affecting application performance.
  - Providing early & cheap performance and cost prediction of applications.

CG Task 2.3: Roadmap

- Definition of a model and a suite of metrics.
- Overview of benchmarks for HPC and HTC: SPEC, Linpack, SPLASH, Parkbench, NAS, etc.
- Consolidation of X#-application kernels.
- Definition of CrossGrid Benchmark Suite (**CGBS**).
- **CGBS** Implementation and Testing.
- Performance Evaluation of benchmark tests.

CGBS Requirements

- Layered X# Architecture & Modular Applications
  - Hierarchical benchmarks.
- Estimating performance capacity (and cost?) of VO components.
  - Need for generic & simple kernels.
- Focusing on X# applications.
  - Need for kernels representative of X# applications.
- Focus on user-friendliness & portals.
  - Web-based access to benchmarks & results.



CGBS Layers

- ***Micro-benchmarks and Generic Kernels***
  - For “stress testing” and identification of basic performance capacity of grid configurations.
  - Generic HPC/HTC kernels.
  - Computing power (flops), Storage/Network Capacity, cost estimate, etc.
- ***Application Kernels***
  - Characteristic of representative CG applications.
  - Higher-level metrics (completion time, throughput, speedup…)
- ***Middleware Kernels***
  - *To investigate the effects of middleware on end-to-end application performance.*
  - *Characterization of middleware performance capacity.*

CGBS Deployment

We anticipate:

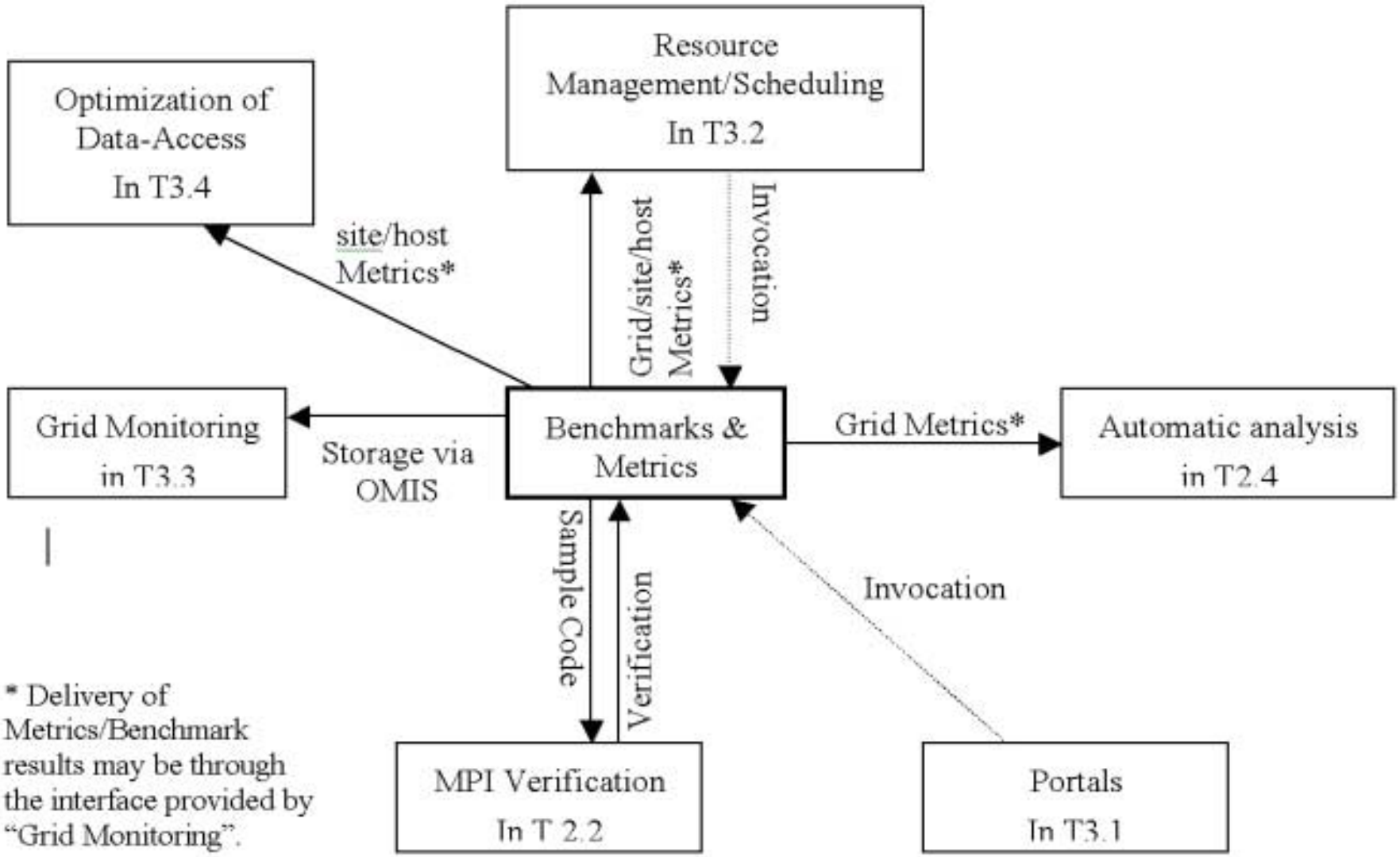
- **Quick** tests
  - Less accurate estimations **quickly.**
  - Frequent invocations (e.g. by a scheduler to quickly determine site performance).
  - Could probably be run at other than idle-time.
- **Thorough** tests:
  - Better suited for a more accurate representation of expected system performance.
  - take longer to run and require more resources.
  - E.g., in the case of new VO construction.

CGBS Operation

- Benchmark invocation
  - By users through portal.
    - Determine Grid/Site/Host performance
    - Estimate Completion time of application and cost
  - By users/administrators through simple interface
    - Site administration/testing
  - By services
    - Such as a scheduler
    - (Grid Monitoring perhaps)

Interactions with other Tasks

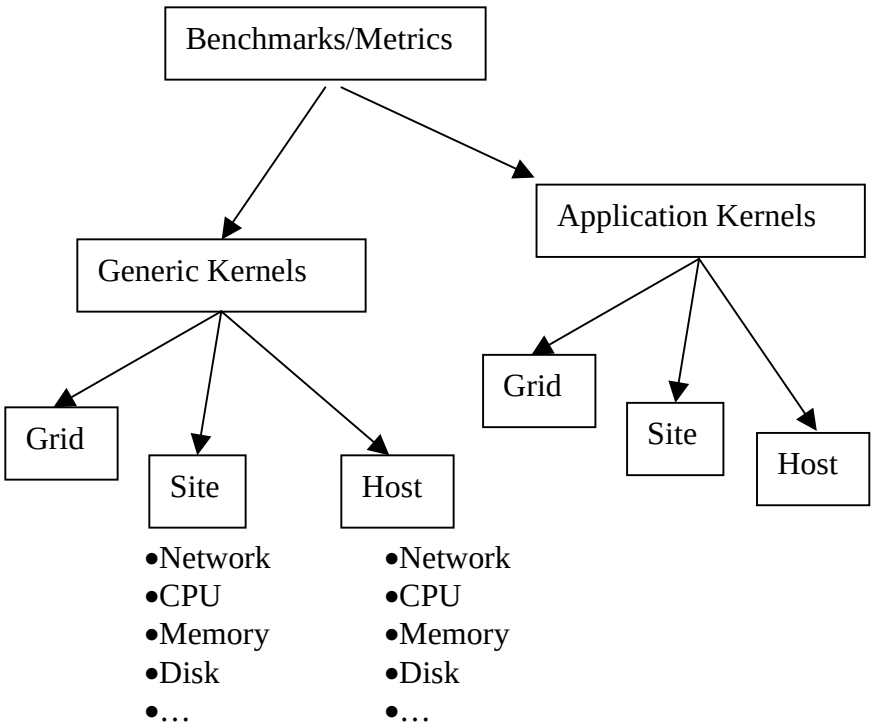
high-level schema



Benchmark data

- Publication
  - via T3.2 ?
  - through GIIS ? (i.e. treat it as a site property, makes sense for a resource manager)
- Storage of historical benchmark data (necessary?) in collaboration with T3.2.
  - (Should T2.3 talk OMIS?)

Benchmark targets



- Typical benchmarks will cover (at Grid/Site/Host levels)
  - CPU performance
  - Memory performance/bandwidth
  - Disk (Data Access Opt. T3.4?)
  - Network
- Existing open-source benchmarking code will be used as much as possible
  - (Linpack, e.t.c.)

T2.3 Timeline

M1-3 Requirements definition phase, Overview

M4-6 Model, Metrics & Benchmark definition

M7-12 First prototypes - CGBS v1.0

M13-14 Integration with WP4

M15-18 Benchmark Suite Refinement – CGBS v1.1.1

M19-24 Implement fully Grid-enabled prototypes – CGBS v1.1.2

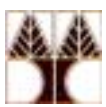
M25-33 Benchmark evaluation/refinement – CGBS v1.2

M33-36 Finishing phase

Questions & Issues

Foundations for WP2:

- Seeking *realistic* X#-application models representing the deployment of X# applications by the end of CrossGrid:
  - C(C++/FORTRAN)+MPI-G2 expanding over different Grid nodes?
  - Interacting, loosely-coupled modules?
  - Both?
- Need a structured approach for consolidating:
  - End-user requirements for WP2 tools.
  - Kernels of X# applications.
  - Workload descriptions and input data.



Questions to address

Intra-WP2 interactions and overlaps:

- Application kernel consolidation: subject of both Tasks 2.3 and 2.4.
- Common reference model for a Grid architecture.
- Relation between T2.3 Metrics & models, and performance properties and analytical models of T2.4.
- Performance measurement of the CGBS.
- CGBS and performance prediction.
- CGBS as an evaluation vehicle for automatic analysis.

More questions to address

Interactions of CGBS with WP3 and DataGrid:

- Is a model for the middleware needed? Who will provide this?
- Consolidation of middleware services affecting application performance and subject to benchmarking. When are they going to be ready?
- Relation with Performance Monitoring? Will CGBS be used as an accuracy test for Perf. Monitoring or is Perf. Monitoring going to be used for extracting CGBS metrics?
- How are we going to get the resources made available for a particular application?

Other Issues

Testing and Deployment of CGBS:

– Which X# nodes will be available for early tests and prototypes?

Mechanisms for collaboration:

– Working meetings.

– Short visits.

– Telephone and VRVS-netmeeting conferences.