Design and Implemention of a Plugin Scheduler for DIET & Performance Prediction in DIET with CoRI Collectors of Resource Information

February 19, 2006

Outline

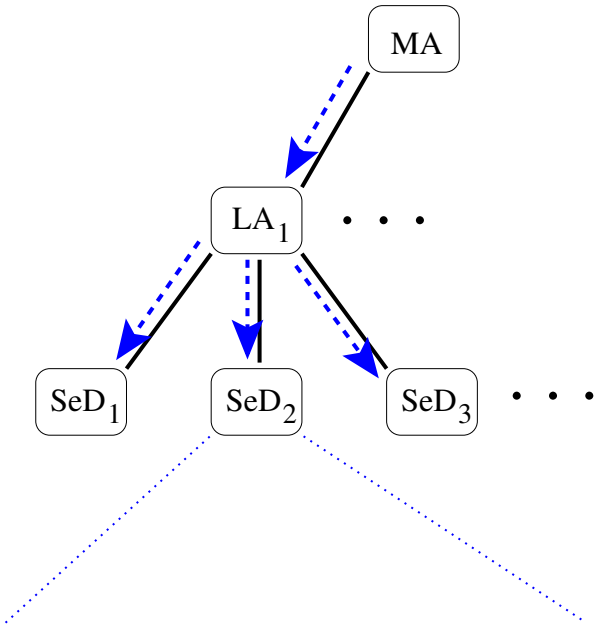
Outline

Grids and DIET

- Grid platforms
  - Heterogeneous computational resources
  - Irregular network topologies
  - Dynamic resource performance
- DIET philosophy and design principles
  - Server and broker agent model
  - Hierarchical organization
  - Flexible deployment options

DIET Overview

DIET hierarchy:



SeD response (DIET_profile_t)

COMP_TIME	t_{comp}
COMM_TIME	t_{comm}
TOTAL_TIME	t_{total}
AVAIL_MEM	m_{avail}

Basic progress of a DIET call:

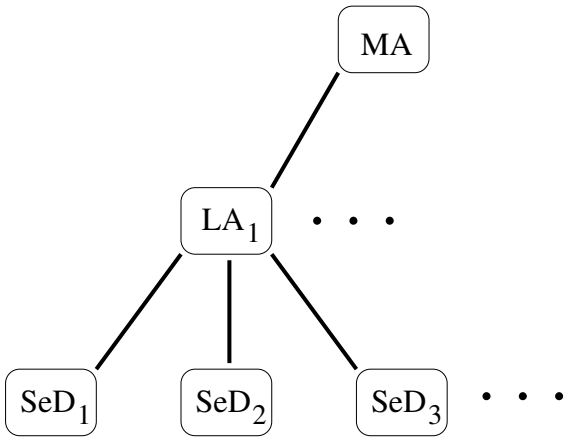
- Client requests service from the Master Agent (MA)
- The MA interrogates the DIET hierarchy
- Each Server Daemon (SeD) responds with a response profile
- Each Local Agent (LA) compiles and sorts the responses by execution time
- MA returns a list of servers to the client
- Client launches service directly on SeD

Problem

- Non-standard application- and platform-specific performance measures

Application-specific Performance Use Case

DIET hierarchy:

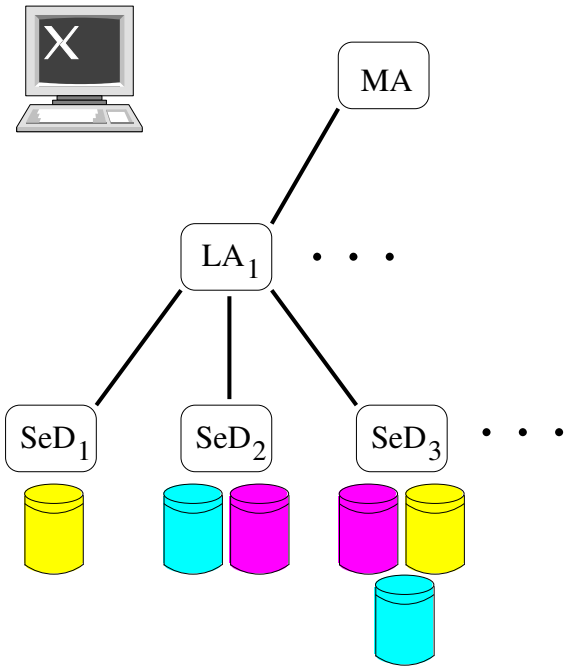


Motivation

- Basic DIET deployment

Application-specific Performance Use Case

DIET hierarchy:

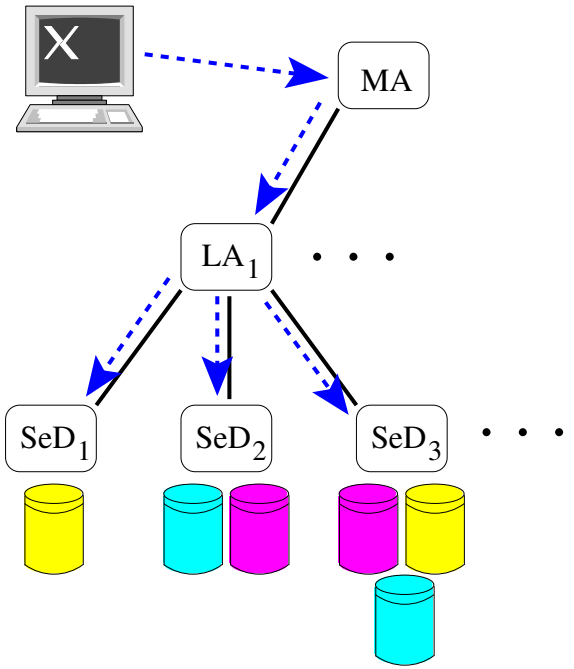


Motivation

- Basic DIET deployment
- Client application with data dependencies

Application-specific Performance Use Case

DIET hierarchy:

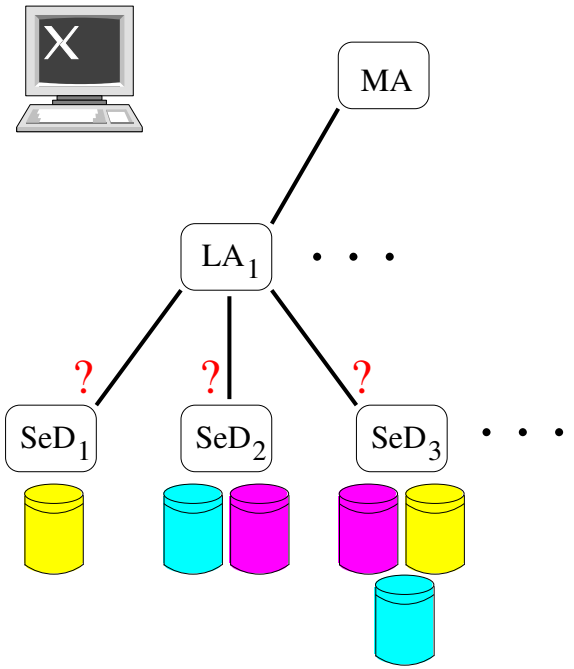


Motivation

- Basic DIET deployment
- Client application with data dependencies

Application-specific Performance Use Case

DIET hierarchy:

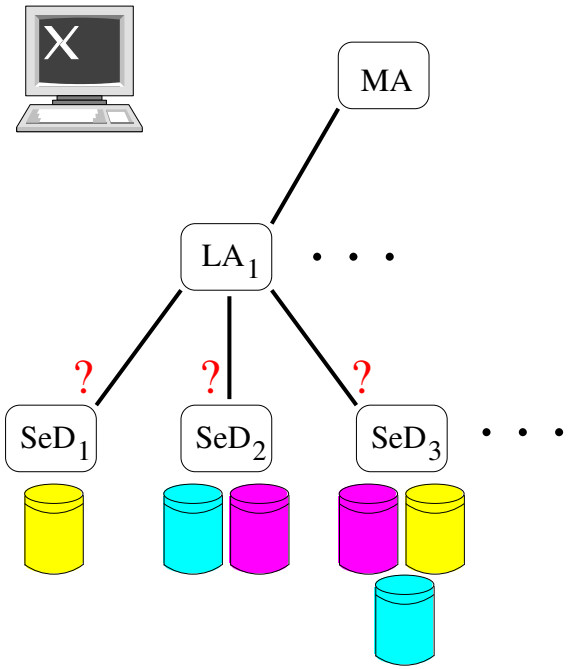


Motivation

- Basic DIET deployment
- Client application with data dependencies
- "performance" is not well-defined

Application-specific Performance Use Case

DIET hierarchy:



Motivation

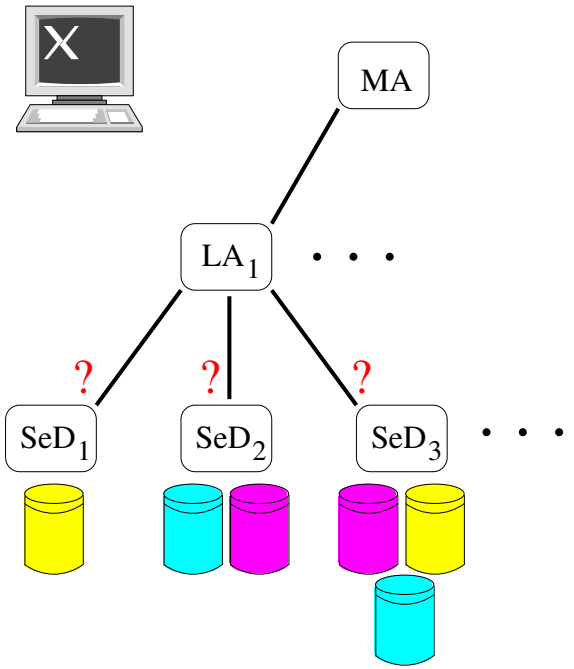
- Basic DIET deployment
- Client application with data dependencies
- "performance" is not well-defined

Possible meanings for performance

- Existence of data
- Avail. free memory
- Specific architecture
- Previous scheduling decisions
- Application-specific measures
- Composite requirements
- ...

Application-specific Performance Use Case

DIET hierarchy:



Motivation

- Basic DIET deployment
- Client application with data dependencies
- "performance" is not well-defined

Possible meanings for performance

- Existence of data (GriPPS)
- Avail. free memory (MUMPS?)
- Specific architecture (TLSE)
- Previous scheduling decisions
- Application-specific measures
- Composite requirements
- ...

Context-sensitive performance metrics are needed

Outline

Plugin Scheduling

Plugin scheduling facilities to enable

- Application-specific definitions of appropriate performance metrics
- An extensible measurement system
- Tunable comparison/aggregation routines for scheduling

Plugin Scheduling

Plugin scheduling facilities to enable

- Application-specific definitions of appropriate performance metrics
- An extensible measurement system
- Tunable comparison/aggregation routines for scheduling

Design changes

Component	Before	After
SeD	automatic performance estimate (FAST/NWS)	chosen/defined by application programmer

Plugin Scheduling

Plugin scheduling facilities to enable

- Application-specific definitions of appropriate performance metrics
- An extensible measurement system
- Tunable comparison/aggregation routines for scheduling

Design changes

Component	Before	After
SeD	automatic performance estimate (FAST/NWS)	chosen/defined by application programmer
Agents	exec. time sorting	"menu" of aggregation methods

Plugin Scheduling

Plugin scheduling facilities to enable

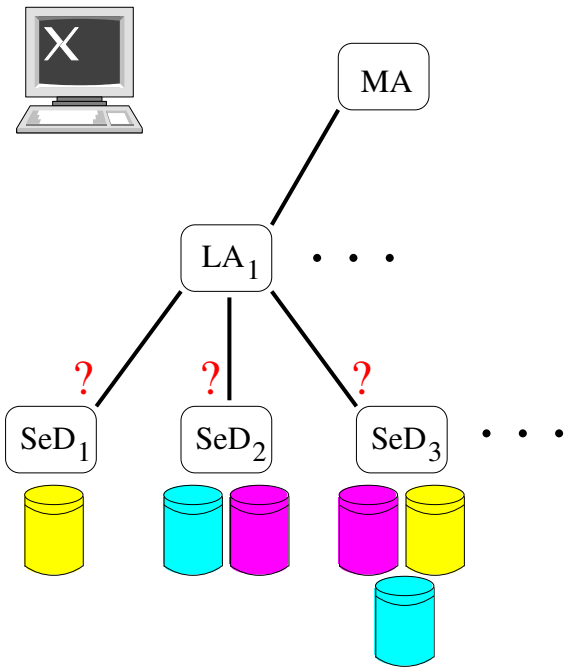
- Application-specific definitions of appropriate performance metrics
- An extensible measurement system
- Tunable comparison/aggregation routines for scheduling

Design changes

Component	Before	After
SeD	automatic performance estimate (FAST/NWS)	chosen/defined by application programmer
Agents	exec. time sorting	"menu" of aggregation methods
Client	CLIENT CODE UNCHANGED	

Plugin Scheduling Enhancements

DIET hierarchy:

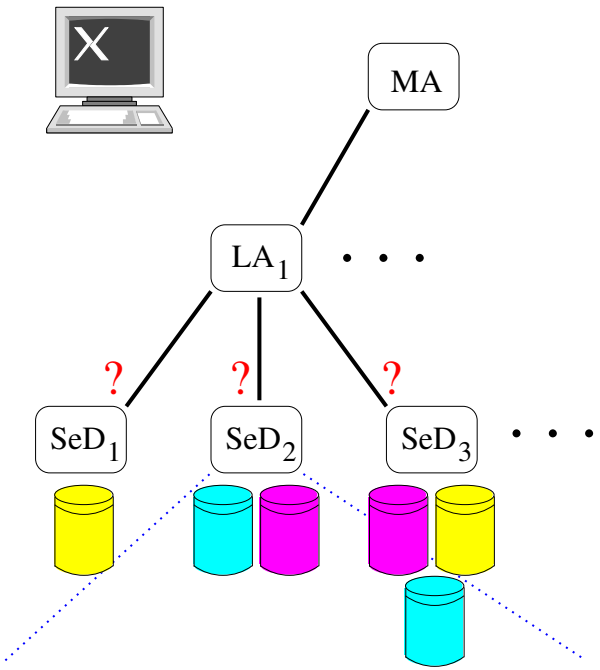


Example: Client request for comparison operation on blue database or Juxmem repository

- Request arrives at SeD level

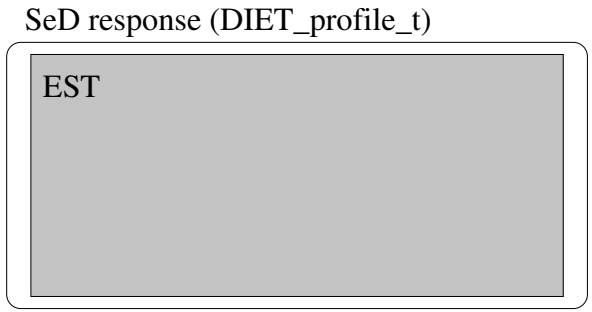
Plugin Scheduling Enhancements

DIET hierarchy:



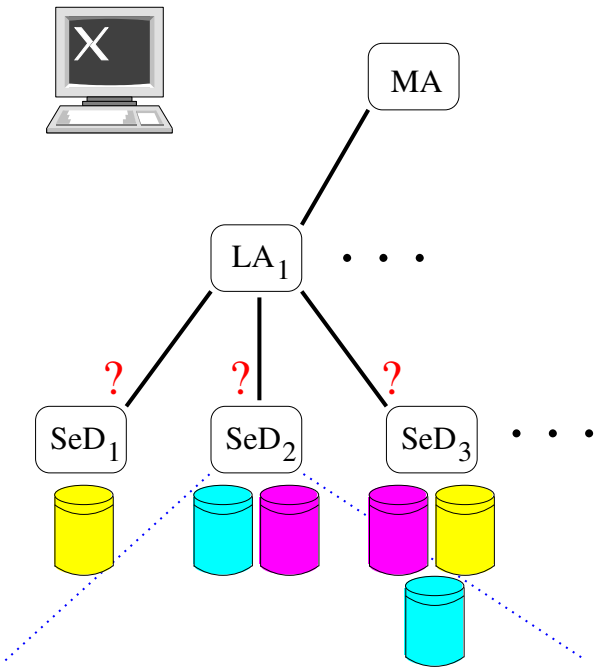
Example: Client request for comparison operation on blue database or Juxmem repository

- Request arrives at SeD level
  - Profile is an *esimation vector*: dynamic array of (tag,value) pairs



Plugin Scheduling Enhancements

DIET hierarchy:



Example: Client request for comparison operation on blue database or Juxmem repository

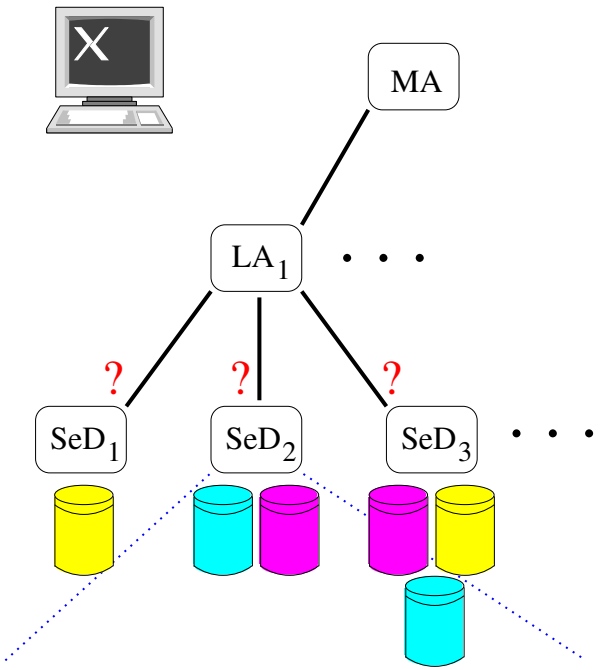
- Request arrives at SeD level
  - Profile is an *esimation vector*: dynamic array of (tag,value) pairs
  - Contains standard performance metrics

SeD response (DIET_profile_t)

EST		
	COMP_TIME	t_{comp}
	COMM_TIME	t_{comm}
	AVAIL_MEM	m_{avail}

Plugin Scheduling Enhancements

DIET hierarchy:



Example: Client request for comparison operation on blue database or Juxmem repository

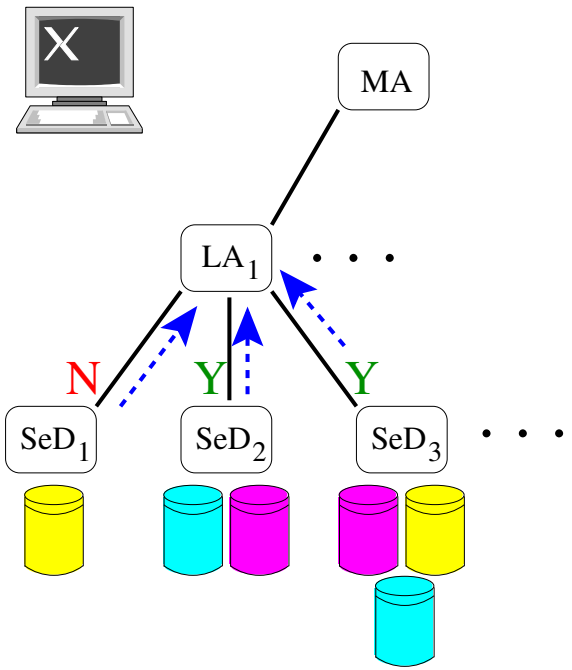
- Request arrives at SeD level
  - Profile is an *esimation vector*: dynamic array of (tag,value) pairs
  - Contains standard performance metrics *and* application-specific data

SeD response (DIET_profile_t)

EST	COMP_TIME	t_{comp}
	COMM_TIME	t_{comm}
	AVAIL_MEM	m_{avail}
	DB_BLUE	Y
	DB_YELLOW	N
	DB_PURPLE	Y

Plugin Scheduling Enhancements

DIET hierarchy:

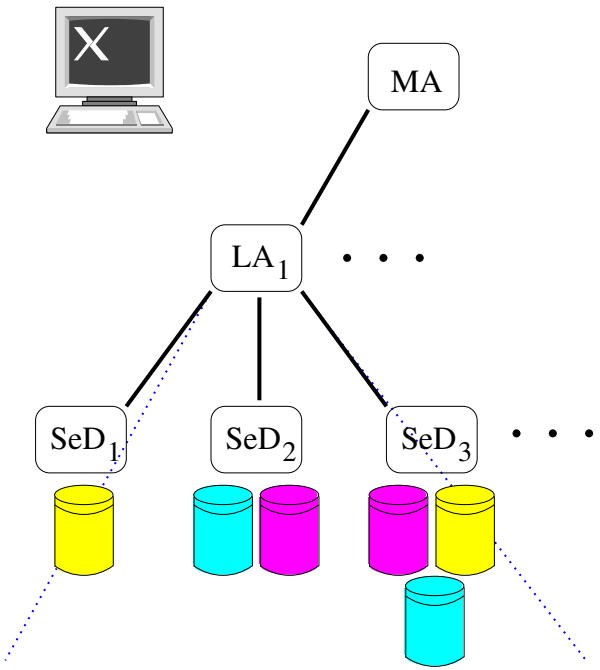


Example: Client request for comparison operation on blue database or Juxmem repository

- Request arrives at SeD level
  - Profile is an *esimation vector*: dynamic array of (tag,value) pairs
  - Contains standard performance metrics *and* application-specific data
- Custom responses propagated up

Plugin Scheduling Enhancements

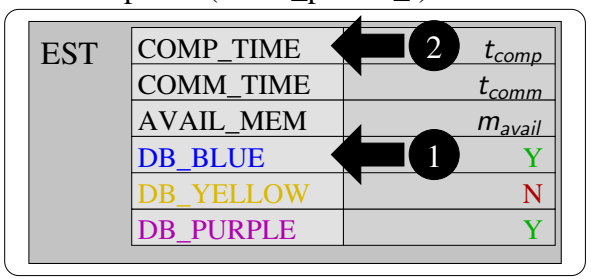
DIET hierarchy:



Example: Client request for comparison operation on blue database or Juxmem repository

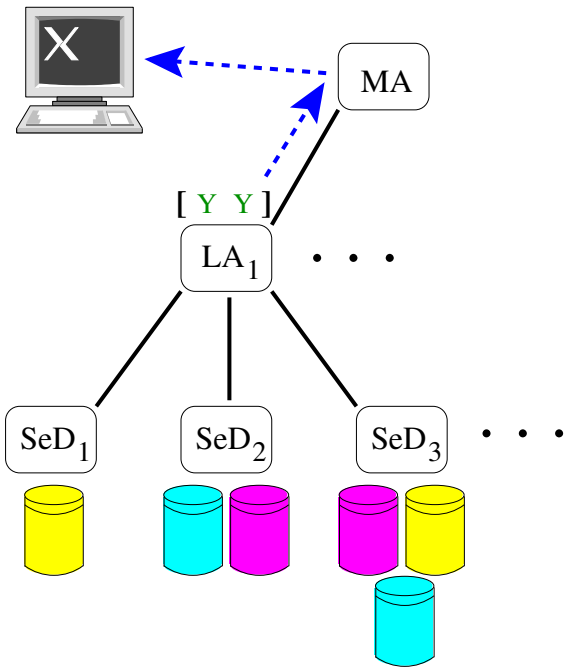
- Request arrives at SeD level
  - Profile is an *esimation vector*: dynamic array of (tag,value) pairs
  - Contains standard performance metrics *and* application-specific data
- Custom responses propagated up

SeD response (DIET_profile_t)



Plugin Scheduling Enhancements

DIET hierarchy:



Example: Client request for comparison operation on blue database or Juxmem repository

- Request arrives at SeD level
  - Profile is an *esimation vector*: dynamic array of (tag,value) pairs
  - Contains standard performance metrics *and* application-specific data
- Custom responses propagated up
- Enables various selection methods
  - Basic resource availability
  - Processor speed, memory
  - Database contention
  - Future requests

Implementation Mechanisms

What mechanisms are needed to implement this framework?

- **SeD-level** (response to client request)
  - Interrogate the system performance
  - Store selected performance metrics
- **Agent-level** (aggregation of server responses)
  - Collect server responses and extract stored performance estimates
  - Order responses from children, based on provided metrics
  - Forward ordered responses to next higher level

SeD-level Interface

Estimation Vector

- Dynamic array of *estimation values*:
  - tag (byte) + value (float)
  - `estVector_t new_estVector()`
- Accessing the EstVector
  - int diet_est_set(estVector_t ev, int userTag, double value);
  - double diet_est_get(estVectorConst_t ev, int userTag, double errVal);
  - double diet_est_get_system(estVectorConst_t ev, int systemTag, double errVal);
- Tags and access functions for existing performance metrics
  - FAST/NWS (e.g, `int diet_estimate_fast(estVector_t, const diet_profile_t*)`)
  - SeD execution timestamp (to approximate Round-robin scheduling)
  - CoRI (EST_NBCPU, EST_CPUSPEED, EST_FREEMEM, EST_FREESIZEDISK, EST_DISKACCESSREAD,..)

Agent-level Interface

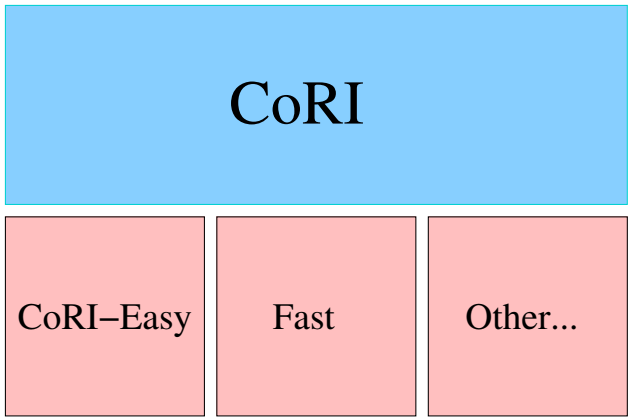
New Profile Parameters

- New dynamic array of prioritized *optimization directives*:
  - *tag*: basis for comparison
  - *semantics*: maximize, minimize, etc.
- At service registration time, directives are fixed
- At runtime, directives used to order server responses

Outline

Presentation

- Collector: an easy interface to gather performances on a SeD
  - Load, memory, disk, network, ...
- Currently, 2 modules supported: CoRI-Easy and Fast
- Can be extended: Ganglia, MDS, ...



API

- type_collector={
  EST_COLL_EASY,
  EST_COLL_FAST}
- Some functions
  - int diet_estimate_cori(
    estVector_t ev,
    int info_type,
    diet_est_collect_tag_t collector_type,
    void* data);
  - int diet_estimate_cori_add_collector(
    diet_est_collect_tag_t collector_type,
    void* data);

CoRI-Easy

- Using fast and basic functions or simple performance tests
- Keep the independence of DIET
- Able to run on "all" operating systems to allow a default scheduling with basic information

Taking the previous example of the blue database...

The code would look like:

```
void set_up_scheduler(char *schedulertype, diet_profile_desc_t*
profile){
        diet_estimate_cori_add_collector(EST_COLL_EASY,NULL);
        diet_aggregator_desc_t *agg;
        agg = diet_profile_desc_aggregator(profile);
        diet_service_use_perfmetric(performance_Load_Avg);
        diet_aggregator_set_type(agg, DIET_AGG_PRIORITY);
        diet_aggregator_priority_min(agg, EST_AVGFREECPU);
}

void performance_Mem_Free(estVector_t perfValues){
        diet_estimate_cori(perfValues,
                           EST_FREEMEM,
                           EST_COLL_EASY,
                           NULL);
}
```

Outline

Conclusion & Future Works

Conclusions

- Plugin schedulers available in DIET
- A unified perf. prediction CoRI
- Perf. prediction modules: CoRI-Easy, Fast

Future Works

- Improve CoRI-Easy: faster and better
- Implement other collectors, like Ganglia
- Improve the default DIET scheduling algorithm