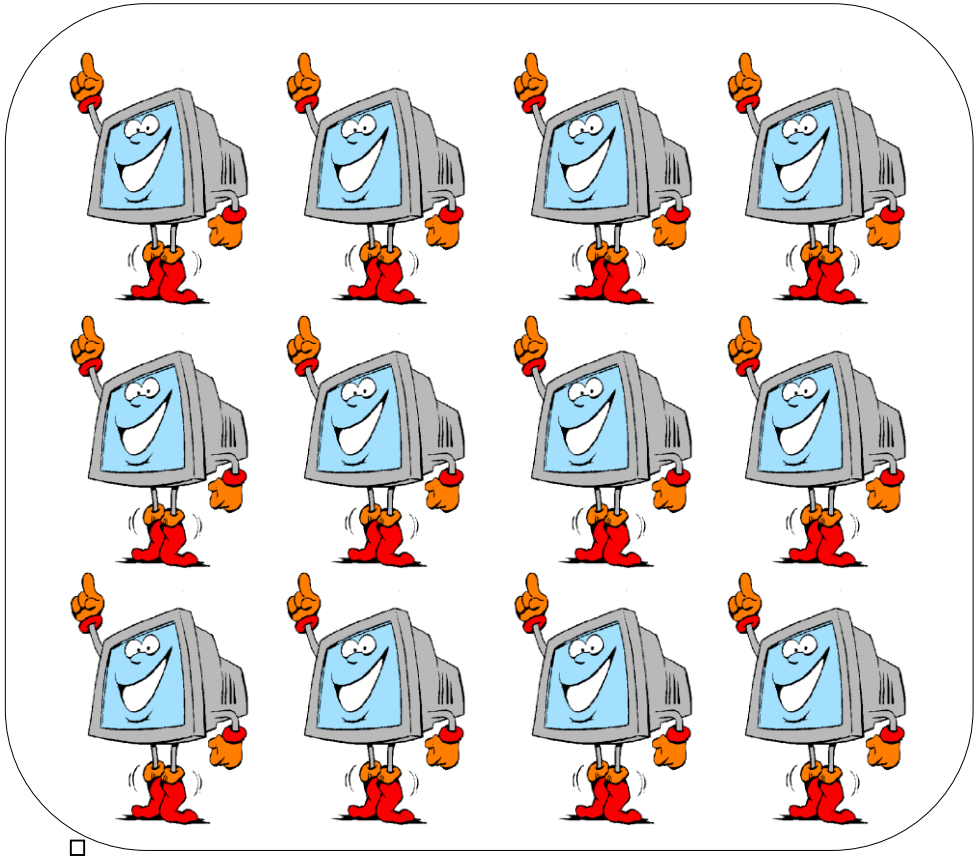
Gestione della convivenza dei job paralleli e sequenziali con torque in UniNa

L.Carracciuolo

G.B.Barone, V.Boccia, D.Bottalico

Outline

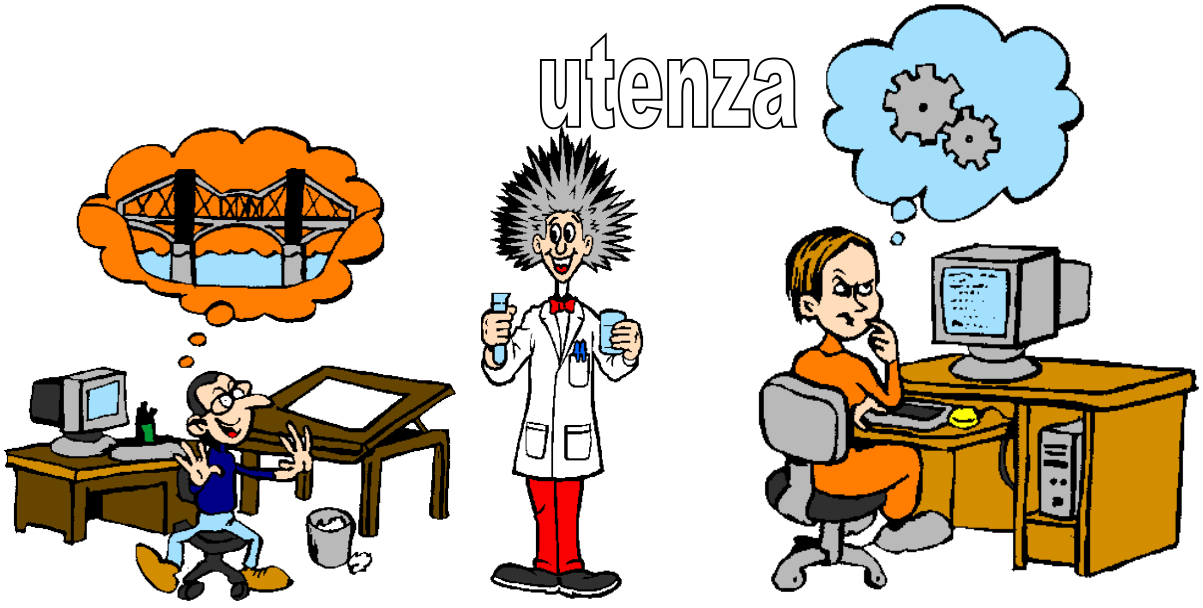
- Il problema: condivere la stessa risorsa ad un'utenza diversificata (HTC/HPC)
- La soluzione al problema
  - gli attori: le risorse, l'utenza e il gestore di risorse
  - Dettagli implementativi
- Conclusioni e sviluppi futuri



risorse

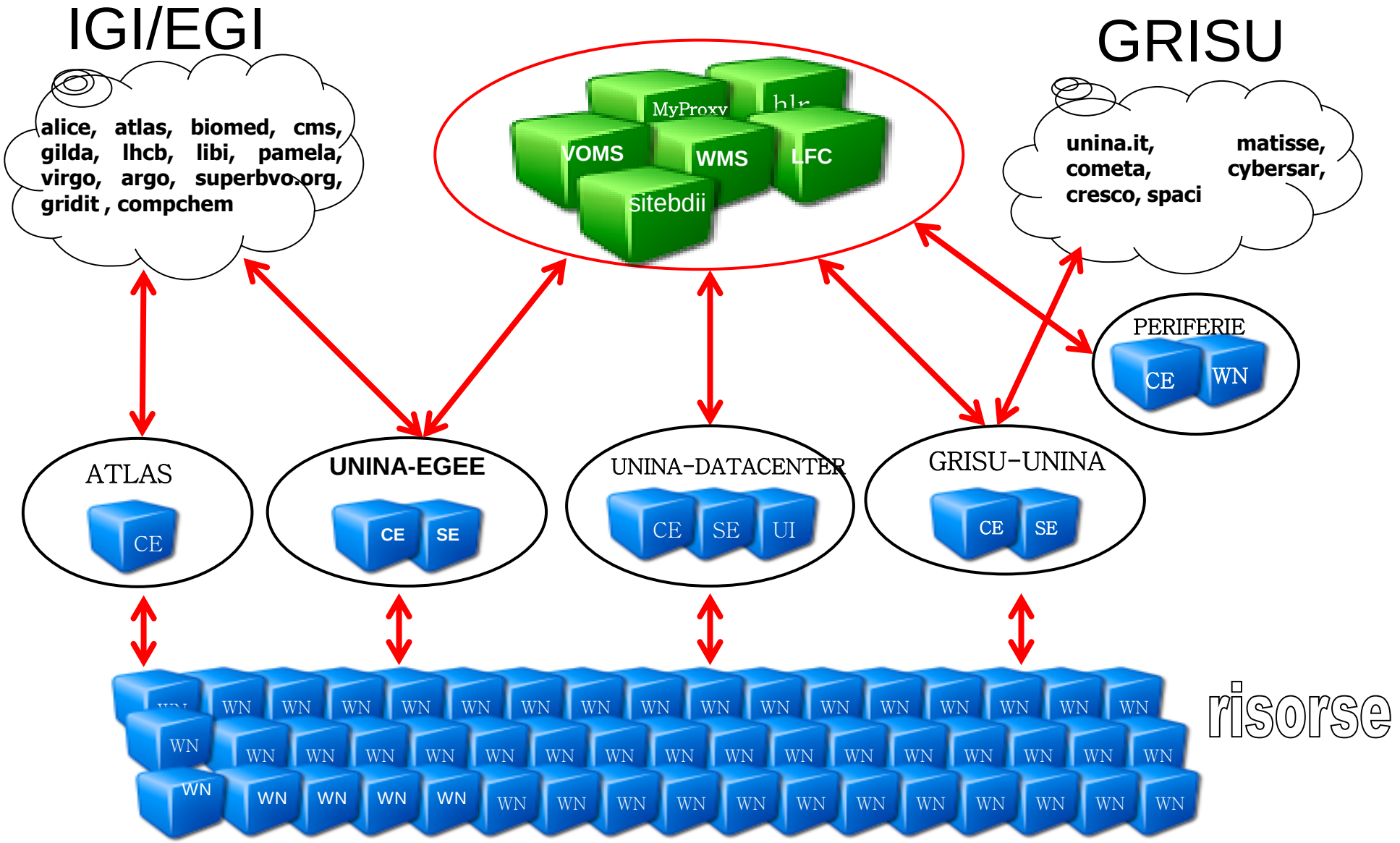
Gestore delle risorse & Scheduler

utenza



La realizzazione di sistemi di calcolo di grosse dimensioni comporta enormi investimenti ed elevati costi di gestione. Tali sistemi hanno il compito di soddisfare le esigenze di un grande, e spesso eterogeneo, bacino di utenza e di fornire, a valle dell'investimento fatto, il massimo ritorno a chi li utilizza.

L'infrastruttura di UNINA



- General purpose
- Dinamica e adattabile all'esigenze dell'utenza diversificata

L'utenza

Tipologie di applicazioni:

- Applicazioni tradizionali in ambito IGI/EGI (High Energy Physics, …)
- Applicazioni HTC (High Troughput Computing) (altre comunità ad es. Medicina, Bioinformatica)
- Applicazioni HPC (High Performance Computing) (Ingegneria es: simulazioni di fluidodinamica effettuate mediante codice commerciale, open o autoprodotto)

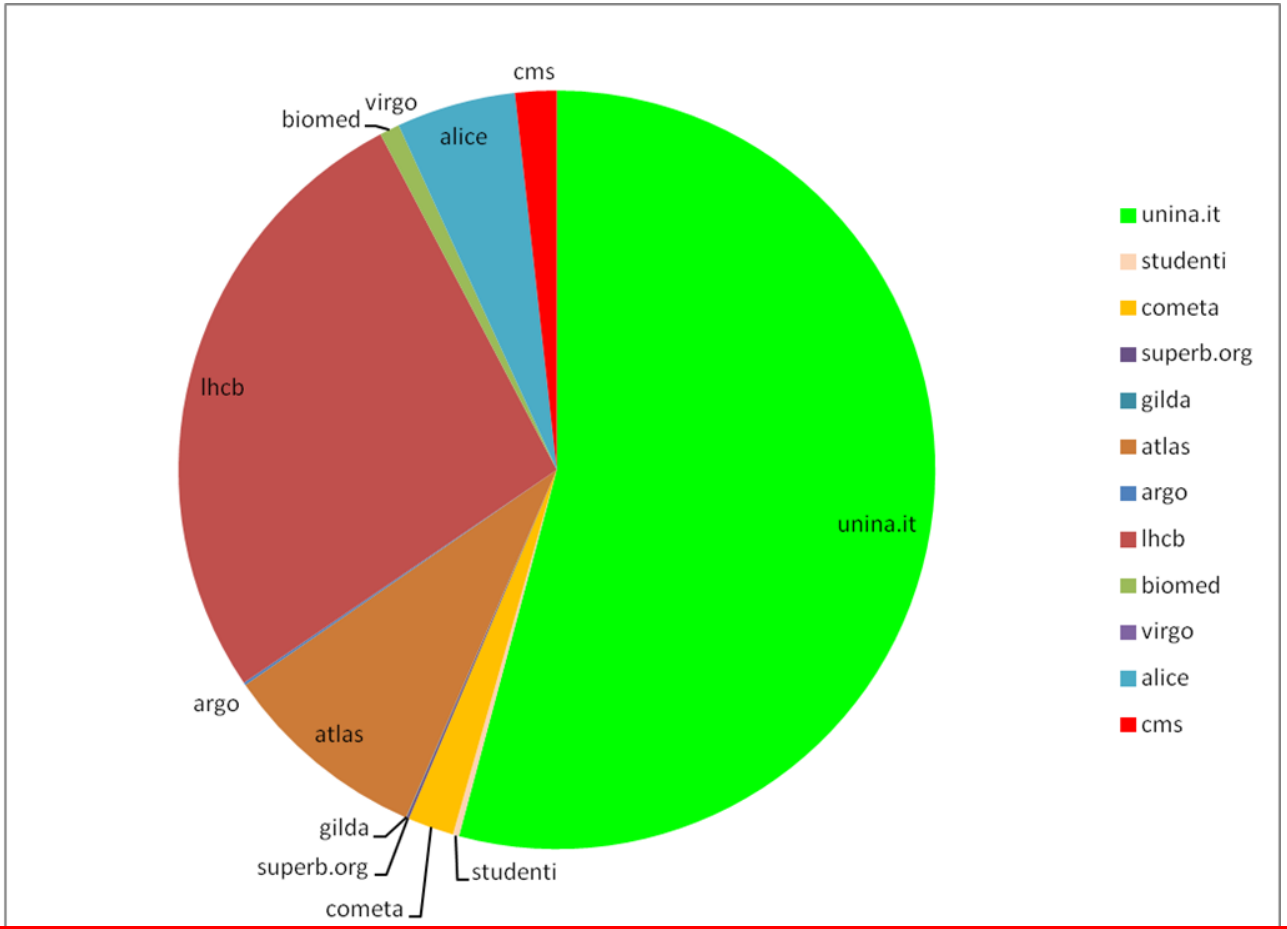
Comunità:

- Comunità scientifiche (di Ateneo e non)
- Studenti (di Ateneo e non)

L'utenza "globale"

Logicalmente classificabile in tre macrogruppi:

- **GRISU`**: appartenenti alle Virtual Organizzation (VO) *cometa*, *cybersar*, *cresco* e *spaci*
- **IGI/EGI**: appartenenti alle VO *alice, atlas, biomed, cms, gilda, lhcb, libi, pamela, virgo, argo, superbvo.org, gridit, compchem*
- **utenza LOCALE**: appartenente alle VO *unina.it*, *matisse*

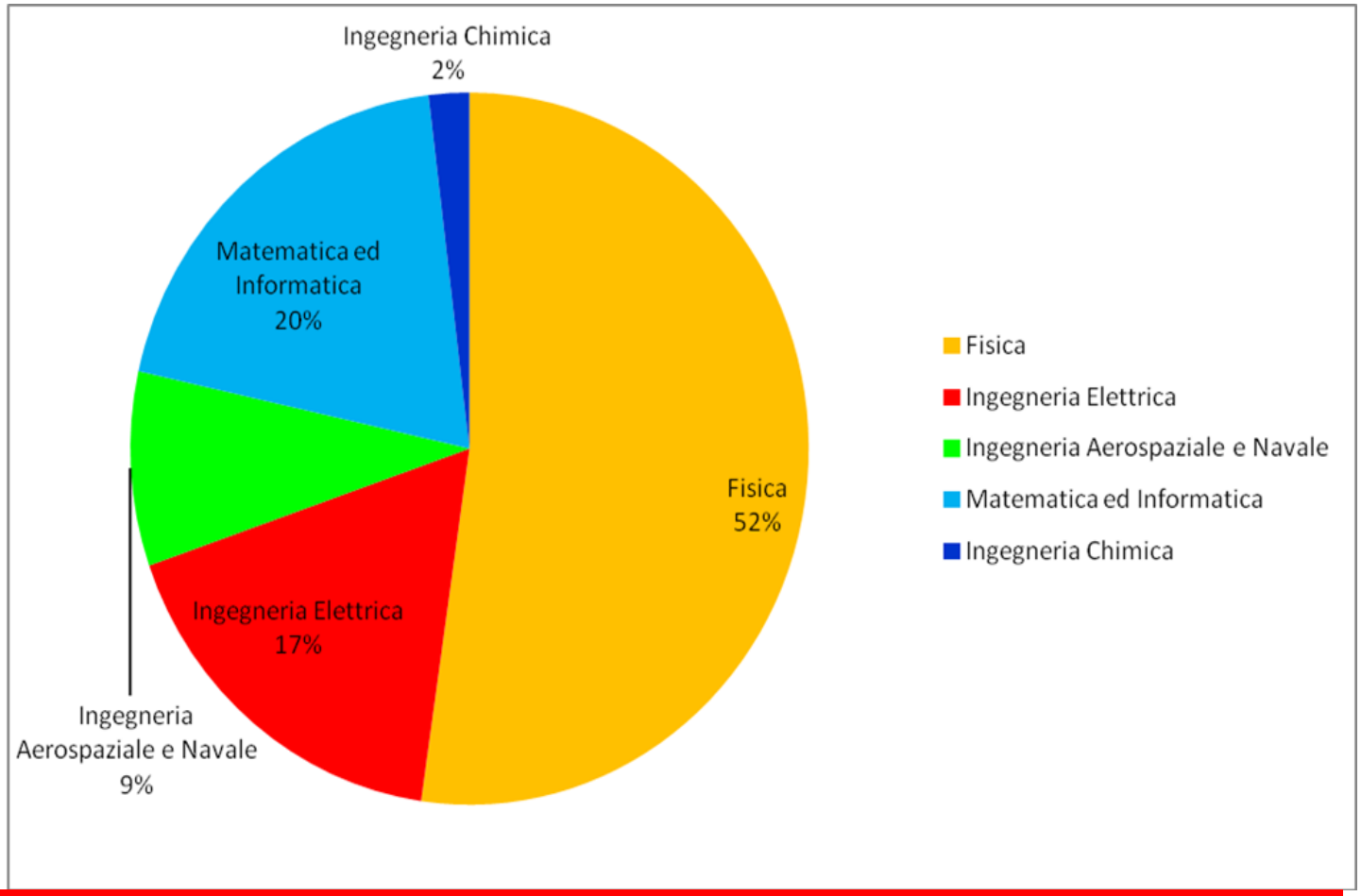


Suddivisione del tempo macchina utilizzato nel secondo semestre del 2011

L'utenza locale

I grandi utenti della VO *unina.it* e *matisse*

- Astrofisica
- Chimica
- Fisica della materia
- Fisica subnucleare
- Medicina/Bioinformatica
- Matematica/Informatica
- Ingegneria aerospaziale
- Ingegneria chimica
- Ingegneria elettrica
- Ingegneria informatica
- Ingegneria navale



Suddivisione del tempo macchina utilizzato nel secondo semestre del 2011

Il gestore delle risorse di UNINA

Le politiche di accesso implementate sono:

- gestite da un sistema basato sulla combinazione Torque/Maui,
- basate su un insieme di code definite sulla base dell'appartenenza ai macrogruppi di utenza, della durata e del tipo di job:

una coda di certificazione: egeecert

code GRISU': grisu_short, **grisu_long**: punto di accesso per le VO *cometa*, *cresco*, *spaci*, *cybersar* oltre che per le VO di IGI/EGI

code IGI/EGI:

- **egee_short, egee_long**: punto di accesso per le VO di IGI/EGI
- **atlas**: punto di accesso, per la VO *atlas*, ad un set di 24 nodi ciascuno con connettività FB

code locali:

- **unina_short**, **unina_long**, **unina_infinite**: punto di accesso, per le VO *unina.it* e *matisse*, a 280 nodi ciascuno con connettività IB
- **unina_hpc**: punto di accesso, per job paralleli e sequenziali delle VO unina.it e matisse, ad 80 nodi ciascuno con connettività IB e 16 GB di RAM.

Come ottenere un utilizzo "*armonico*" (bilanciato, efficiente ed efficace) delle risorse ?

Grazie ad opportuna combinazione di:
fairshare, reservation, preemption* e *backfill

- ***fairshare*** per un equo e bilanciato acceso alle risorse
- ***reservation*** per garantire in ogni momento la disponibilità di risorse (es. job di certificazione)
- ***backfill*** per "massimizzare" l'utilizzo di tutte le risorse.
- ***preemption*** per non penalizzare alcune tipologie di job (es. applicazioni non sequenziali) che per loro natura richiedono disponibilità contemporanea di un grande numero di nodi di calcolo e per i quali è quindi più difficile l'allocazione di risorse

Il meccanismo di *fairshare* (1/2)

Tramite il meccanismo del *fairshare* è possibile ottenere un bilanciamento dell'assegnazione delle risorse in funzione di informazioni storiche di utilizzo. Se configurato, il meccanismo di *fairshare* modifica la modalità con la quale viene calcolata la priorità dei job grazie a grandezze, dette "pesi", associate a ciascuna delle entità presenti nella configurazione (utenti, gruppi, code, account).

Il *fairshare* è configurabile mediante due gruppi di parametri. Il primo contribuisce a definire le modalità con le quali lo scheduler MAUI raccoglie e analizza le informazioni relative all'utilizzo delle risorse. Il secondo stabilisce come le informazioni raccolte andranno ad influenzare direttamente le priorità dei job.

I parametri del primo gruppo sono i seguenti:

- `FSINTERVAL`, indica la durata delle finestre temporali
- `FSDEPTH`, indica quante finestre dovranno essere conteggiate
- `FSDECAY`, specifica il fattore di decadimento per ogni finestra.
- `FSPOLICY`, indica la metrica utilizzata.

Il sistema di *fairshare* organizza infatti il tempo in un numero di finestre temporali distinte dette *"fairshare windows"* la cui durata e numero sono definiti mediante i parametri FSINTERVAL e FSDEPTH. Il tempo totale di valutazione del *fairshare* è dato quindi dal valore FSINTERVAL*FSDEPTH.

Una volta configurati tutti i parametri delle policy di *fairshare*, è necessario impostare gli obiettivi per ciascun utente, gruppo, account o coda.

`http://www.adaptivecomputing.com/resources/docs/maui/6.3fairshare.php`

Il meccanismo di *fairshare* (2/2)

Il meccanismo è attivato grazie alla seguente configurazione nel file maui.cfg per i parametri del primo gruppo:

```
FSPOLICY                DEDICATEDPS
FSINTERVAL              04:00:00
FSDEPTH                 16
FSDECAY                 0.75
FSWEIGHT                XXX
```

e per i pesi relativi agli utenti e agli account:

```
FSUSERWEIGHT            XXX
FSACCOUNTWEIGHT         XXX
```

Infine gli obiettivi, per ciascuno degli account e degli utenti:

```
USERCFG[DEFAULT] FSTARGET=XXX
ACCOUNTCFG[atlas] FSTARGET=XXX
ACCOUNTCFG[grisu] MAXPROC=498 FSTARGET=XXX
ACCOUNTCFG[local] FSTARGET=XX
ACCOUNTCFG[egee] MAXPROC=768 FSTARGET=XXX
ACCOUNTCFG[mpi] FSTARGET=XXX
```

Il meccanismo di *reservation*

La *reservation* è un meccanismo con il quale lo scheduler MAUI garantisce la disponibilità di singole risorse o set di risorse in un particolare momento e per ciascun job. Lo scheduler infatti, grazie al meccanismo di *reservation*, effettua una sorta di "prenotazione" delle risorse necessarie a ciascun job.

Ogni *reservation* è formata da tre componenti principali: la lista delle risorse da riservare, lo spazio temporale e la access control list (ACL). Meccanismi di *reservation* "statiche" sono attualmente utilizzati per garantire l'esistenza di risorse accessibili dalle code di certificazione grazie alla seguente configurazione:

```
QOSCFG[cert]QFLAGS=USERESERVED:cert
SRCFG[cert] RESOURCES=PROCS:8
SRCFG[cert] TASKCOUNT=XXX
SRCFG[cert] PERIOD=INFINITY
SRCFG[cert] STARTTIME=0:00:00 ENDTIME=24:00:00
SRCFG[cert] CLASSLIST=egeecert
SRCFG[cert] NODEFEATURES=ibnode
CLASSCFG[egeecert] QDEF=cert
```

http://www.adaptivecomputing.com/resources/docs/maui/7.1.5managingreservations.php

Il meccanismo di *backfill*

Il meccanismo di *backfill* tenta di ottimizzare l'utilizzo delle risorse in base a considerazioni sulla priorità e lunghezza stimata dei job.

Lo scheduler MAUI calcola una priorità per ciascuno dei job presenti in ognuna delle code di sottomissione e, in funzione di tale priorità, ordina tali job. Lo scheduler tenta quindi di eseguire il primo job della lista, quello con priorità maggiore.

Siccome a tutti i job, e a tutte le *reservation* ad essa associate, è assegnato una stima del tempo di partenza e del tempo di durata, lo scheduler può determinare i tempi di completamento di tutti i job in coda e, di conseguenza, quali risorse saranno prima disponibili per il job a maggiore priorità.

Abilitando il sistema di *backfill* lo scheduler può eseguire job a minore priorità senza che tali esecuzioni influenzino quelle dei job a maggiore priorità tentando di “riempire” gli slot non ancora necessari per quest'ultimi job.

Il sistema di *backfill* è attivato grazie alla seguente configurazione nel file `maui.cfg`

```
BACKFILLPOLICY[0]                    XXX
```

`http://www.adaptivecomputing.com/resources/docs/maui/8.2backfill.php`

Il meccanismo di *preemption*

Il meccanismo di *preemption* consente a job con minore priorità di essere sospesi per liberare risorse necessarie all'esecuzione di job a priorità più alta.

La gestione di questo meccanismo viene attivata tramite l'uso di opportuni QOS dello scheduler MAUI. Con questa configurazione i job che si possono sospendere (o *preemptee* job) vengono lanciati solo quando esistono risorse libere e possono essere eseguiti solo fino a quando non è necessario eseguire un job classificato *preemptor*; in quel momento il job *preemptee* viene sospeso allo stato di esecuzione raggiunto e le risorse da esso occupate vengono liberate per il job a maggiore priorità. L'esecuzione dei job sospesi sarà ripresa appena le risorse saranno di nuovo disponibili.

Il sistema di *preemption* sulla coda `unina_hpc` è attivato grazie alla seguente configurazione nel file `maui.cfg`:

```
PREEMPTPOLICY SUSPEND

QOSCFG[preemptor] QFLAGS=PREEMPTOR PRIORITY=XXX
QOSCFG[preemptee] QFLAGS=PREEMPTEE
CLASSCFG[unina_hpc] PDEF=mpi QLIST=preemptor:preemptee
```

Per consentire al middleware LCG/gLite di discriminare, relativamente alla coda `unina_hpc`, fra job sequenziali e job paralleli, si è reso necessario effettuare modifiche su alcuni meccanismi che si interfacciano con il sistema di “scheduling/resource manager” (il file `submit_filter`)

http://www.adaptivecomputing.com/resources/docs/maui/8.4preemption.php

Il file `submit_filter`

```
while (<STDIN>) {

.....
 # By default just copy the line.
    $line = $_;

    # If there is a nodes line, then extract the value and adjust it as necessary..
    if (m/#PBS\s+-l\s+nodes=(\d+)\s*$/) {
        $line = process_nodes($1);

        # If the line wasn't empty, then multiple CPUs have been requested.  Mark this as an MPI job.
        if ($line ne '') {
          $line .= "\n#PBS -A mpi\n";
          $ismpi=1;
        }
    }

.....
    if (m/#PBS\s+-q\s+unina_hpc/) {
          $ishpc=1;
    }

.....
    if($ishpc && /^\w/){
        if($ismpi){
            print "\n#PBS -W x=\"QOS:preemptor;PARTITION:mpi;NACCESSPOLICY:SINGLEJOB\"\n\n";
        } else {
            print "\n#PBS -W x=\"QOS:preemptee;PARTITION:mpi;NACCESSPOLICY:SINGLEJOB\"\n\n";
        }
        $ishpc=0;
    }
    print $line;

.....
 }
```

Conclusioni e sviluppi futuri

- La soluzione proposta (attualmente in produzione sull'infrastruttura UNINA)
  - fornisce "pari opportunità" di esecuzione ai job sequenziali e paralleli
  - consente la serena coesistenza di comunità HTC/HPC (feedback positivi)
- Stiamo lavorando per:
  - automatizzare la riconfigurazione delle policy descritte sulla base di classificazione dei job, log analysis e informazioni storiche di utilizzo
  - modificare ulteriormente il `submit_filter` per job di tipologia diversa da MPI (es. SMP, GPU, etc.)