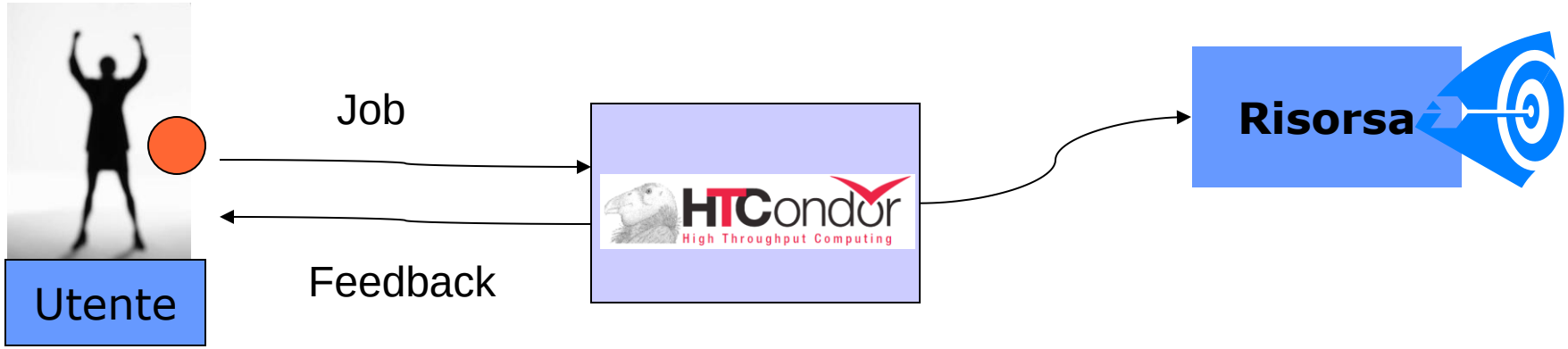
HTCondor

Cosa Fa HTCondor

https://research.cs.wisc.edu/htcondor/



Meccanismi di job management, politica di scheduling, schema con priorità, monitoring di risorse,……

Cosa Fa HTCondor

https://research.cs.wisc.edu/htcondor/

- HT Condor combina un insieme di workstation/server/PC/macchine parallele connesse in rete in una struttura di elaborazione distribuita ad alte prestazioni.
- Condor è basato sulla condivisione collaborativa di risorse di elaborazione.
- Condor utilizza ClassAd Matchmaking per assicurarsi che tutti siano felici.

Filosofia di Flessibilità

- Lasciare il controllo al proprietario
  - Politiche di Uso
  - Decidere quando la risorsa potrà essere usata
  - Proprietari felici -> più risorse ->maggior throughput
- Lasciar crescere naturalmente le comunità
  - Cambi di requisiti e relazioni
  - Contratti non precisi
- Pianificare senza essere prepotenti
  - Non assumere un funzionamento corretto

Sfide di HTCondor

- HT Condor esegue le operazioni necessarie per eseguire i job, anche se alcune macchine ...
  - Sono andate in crash (o si sono disconnesse)
  - Lo spazio su disco di una macchina è esaurito
  - Non c'è il software necessario installato
  - Le macchine sono spesso necessarie ad altri
  - Sono lontane e gestite da qualcun altro.
  - …

Condor: Un Sistema per High Throughput Computing

- Obiettivo
  - Grandi quantità di potenza di elaborazione fault tolerant
  - Utilizzazione effettiva di risorse
- Da ottenere tramite "opportunistic computing"
  - Usa le risorse quando sono disponibili
  - *ClassAds* – per descrivere risorse e jobs
  - Job checkpoint e job migration
  - Remote system calls – preserva l'ambiente di esecuzione locale

HT Condor Kernel

Piano di esecuzione di Jobs

User

Problem Solver

Job

Agent

Matchmaker

ClassAds

Resource

richiesta

Shadow

Dettagli del job

Sandbox

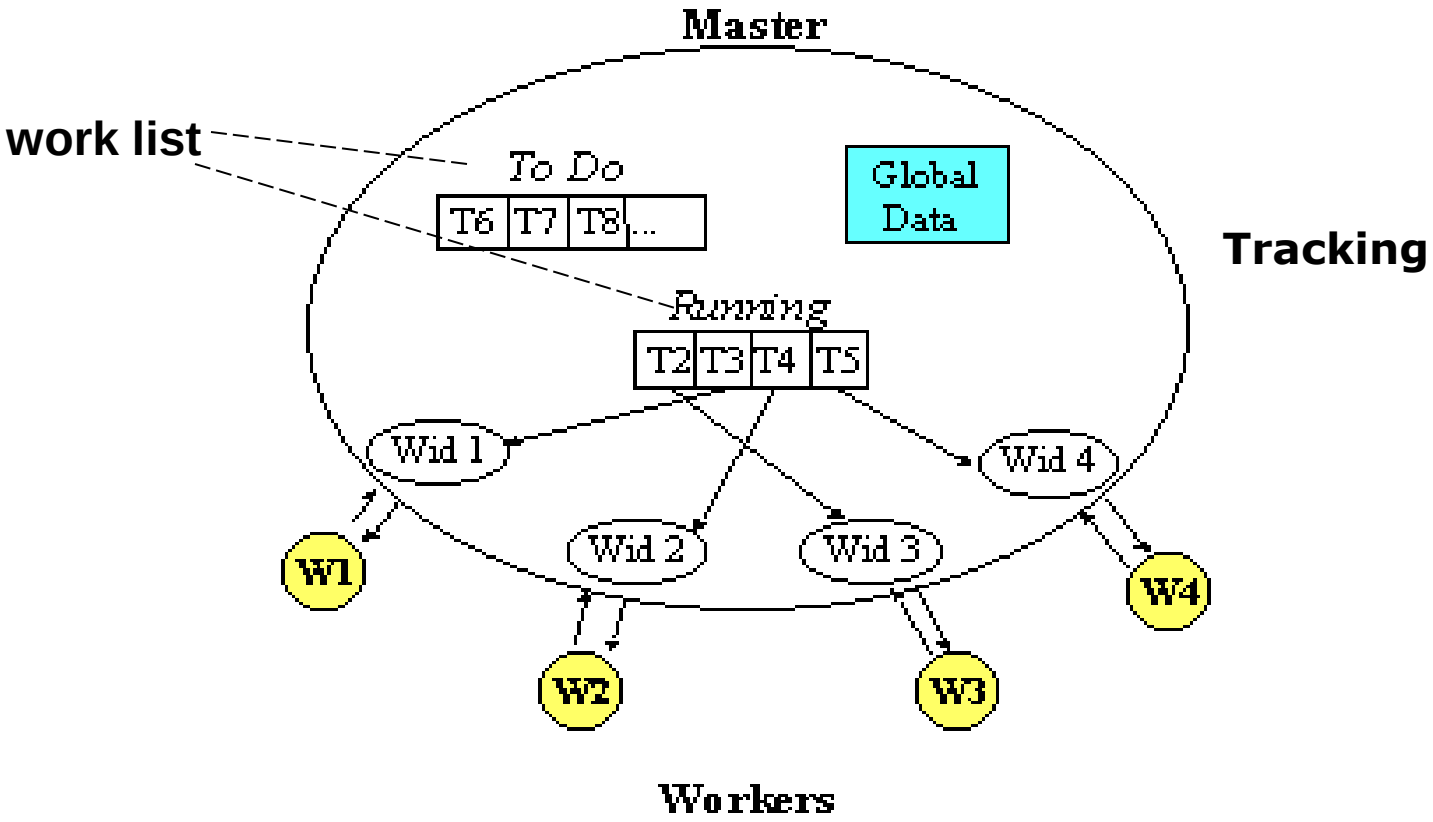
Ambiente

Job

Problem Solver

- Struttura di alto livello costruito ‘sopra’ un Condor agent
- Si occupa dell’ordinamento dei job e della selezione dei task
- È esso stesso rappresentato come un job
  - Un job che sottomette jobs
- Dipende dall’ agent per l’ esecuzione dei task
- Master-Worker, BOSCO e DAG Manager

Master-Worker



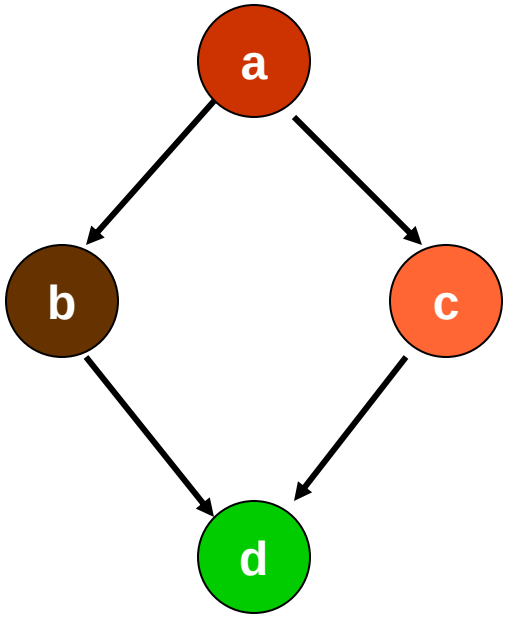
T2 in esecuzione su W3 e monitorato da Wid 3

wisc.edu/condor/mw/june-talk

Directed Acyclic Graph Manager

- Esegue più jobs con dipendenze
- Dipendenze dichiarate usando istruzioni *Parent-Child*
- Programmi speciali da eseguire prima o dopo un job sono specificati da comandi *Pre* e *Post*
  - Per il setup dell'ambiente di esecuzione e l'analisi dei risultati
- L'utente può specificare che un job 'fallito' può essere rieseguito con in comando *Retry*

Directed Acyclic Graph Manager



Job a

Job b

Job c

Job d

Parent a child b c

Parent b child d

Parent c child d

Script Pre c in.pl

Retry c 3

Split Execution

- L'esecuzione di Job richiede
  - Informazione che specifica il job
  - Tool come memoria, rete, ecc.

  Devono essere nello stesso sito
- Shadow
  - Ha informazione che specifica il job
    - Eseguibili, argomenti, file di input, ...
- Sandbox
  - Crea un ambiente per l'esecuzione di un job

Universi Condor

- Universo Condor: Sandbox + Shadow
- Universo Standard
  - Sandbox crea una directory temporanea e la usa per i dettagli sul job in esecuzione
  - Shadow fornisce accesso remoto a device di memoria dell'utente
  - Es. Job richiede un file, la richiesta va allo shadow e il file è memorizzato nel sito di esecuzione.
- Universo Java
- Universo MPI.

Politiche

- Agente
  - Quale risorsa è affidabile?
  - Risorse utili per eseguire jobs
- Risorsa
  - Di quale utente fidarsi?
- Matchmaker
  - Politiche di Comunità, controllo di accesso
- Comunità definite dal matchmaker
  - Agenti possono usare una risorsa solo se condividono un Matchmaker

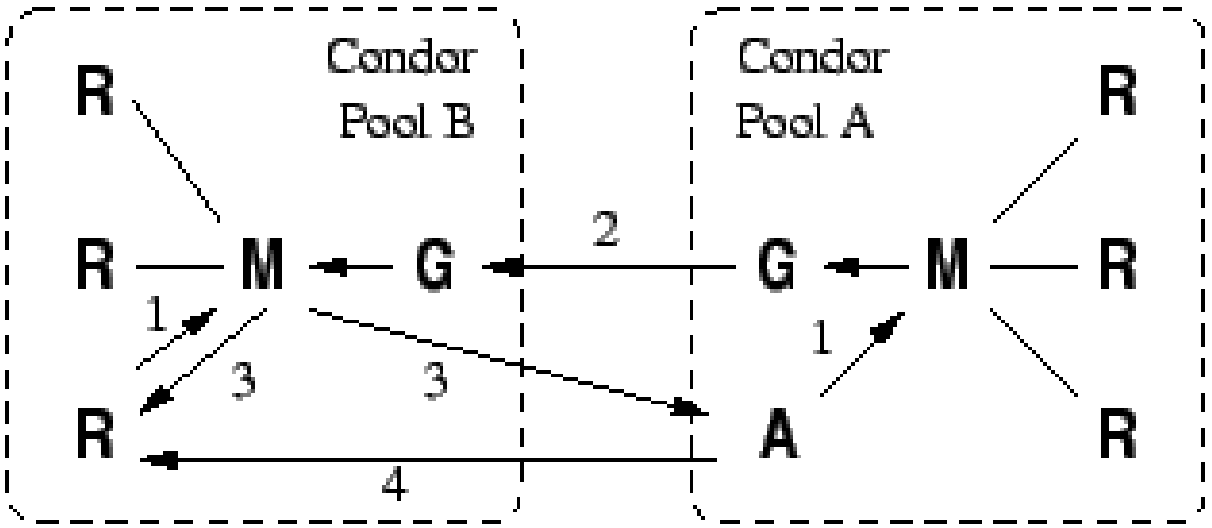
Condor Pool

- Un HTCondor pool è composto da un computer che funge da gestore centrale e da un numero arbitrario di altre macchine.
- Un pool è quindi un insieme di risorse (macchine) che ricevono richieste di uso di risorse (job).
- Ogni risorsa del Condor pool invia aggiornamenti periodici al gestore centrale che le conserva in un repository di informazioni sullo stato del pool.
- Periodicamente, il gestore centrale valuta lo stato attuale del pool e cerca di far corrispondere le richieste in sospeso con le risorse appropriate.
- Un Condor Pool include:

Agent + risorse + Matchmaker (+ Gateway)

Gateway Flocking

Condor Pool: Agent + risorse + Matchmaker (+ Gateway)



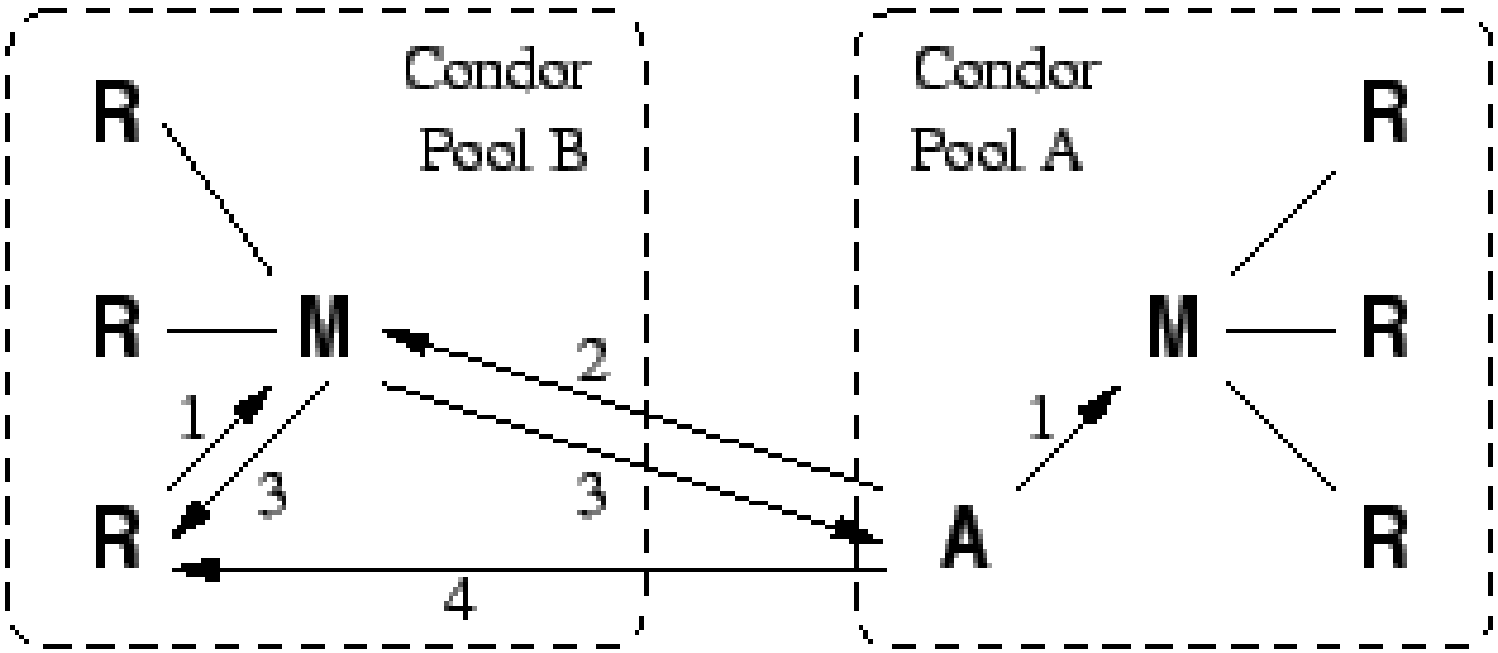
I Gateway passano informazione sui partecipanti tra pool, MA invia richiesta a MB attraverso un gateway G, MB ritorna un *match*

Gateway Flocking

- La struttura dei pool è preservata
- Completamente trasparente : nessuna modifica per gli utenti
- Condivisione a livello organizzativo
- Tecnicamente complesso : i gateway partecipano in tutte le interazioni nel kernel Condor

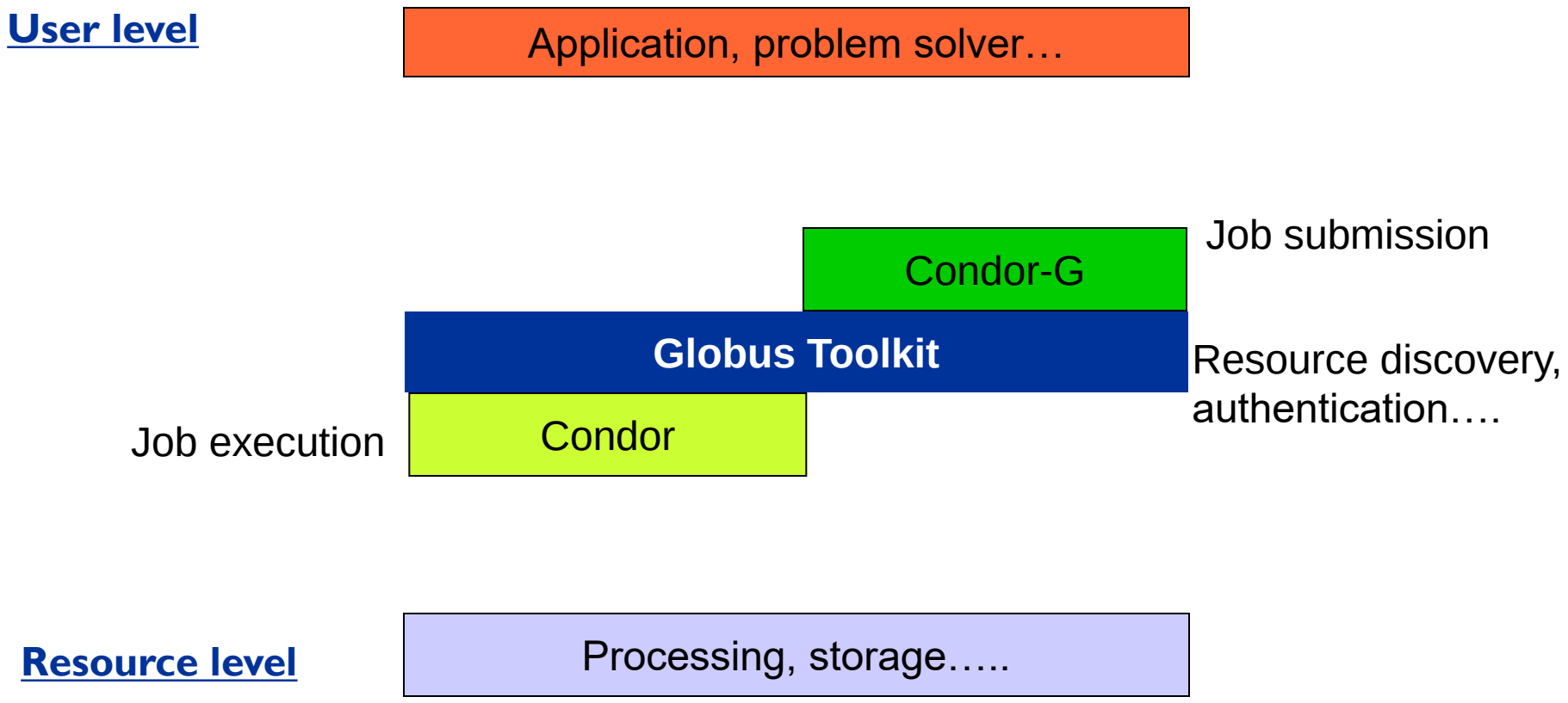
Soluzione: Direct Flocking

Direct Flocking



A interagisce anche con il Condor Pool B

Tecnologie Condor in Middleware di Grid



Matchmaker: Un Ponte tra Planning e Scheduling

- Agenti e risorse pubblicano caratteristiche e requisiti come **ClassAds**
- Sono create coppie che soddisfano i vincoli
- Ambedue le parti sono informate

ClassAds

- Coppie di Attributi nome-valore
- Senza schema specifico
- Logica a tre valori
  - True, false e undefined
- Requisiti
  - Vincoli, per un match devono essere *true*
- Rank
  - Desiderabilità di un match

ClassAds

Job ClassAd

```
[
 MyType = ”Job”
 TargetType =  ”Machine”
 Requirements =
 ((other.Arch== ” INTEL” &&
   other.OpSys== ”LINUX”) &&
   other.Disk > my.DiskUsage)
 Rank = (Memory * 10000)+KFlops
 Cmd = ”/home-exe”
 Department = ”DIMES”
 Owner = ”dtalia ”
 DiskUsage = 6000
]
```

Machine ClassAd

```
[
 MyType= “Machine”
 TargetType=“Job”
 Machine=“tnt.unical.it”
  Requirements=(Load<3000)

  Rank=dept==self.dept
  Arch=“Intel”
  OpSys=“Linux”
 Disk=600000
 ]
```

ClassAds e Entità

Entità	Come visualizzare i classAds
Active Jobs	`$ condor_q -l`
Terminated Jobs	`$ condor_history -l`
Machines (slots)	`$ condor_status -l`
Finished jobs on machine	`$ condor_history -l -file $(condor_config_val STARTD_HISTORY)`
Active submitters	`$ condor_status -submitter -l`
Accounting records	`$ condor_userprio -l`
Schedd service	`$ condor_status -schedd -l`
All services	`$ condor_status -any -l`

ClassAd come *Job* Description

Set di *Attributes*

Attribute:

Key = Value

Key è un nome

Value ha un tipo

```
$ condor_q -l 180.0
```

```
ClusterId = 180
Cmd = "sleep"
DiskUsage = 100
ExitBySignal = false
QDate = 1535384632
RemoteUserCpu = 12.7
RequestDisk = DiskUsage
… (altri attributi)
```

Estensioni al Matchmaking

- **Gang matching**
  - Co-allocazione di più di una risorsa
- **Collections**
  - Memorie persistenti di ClassAds con tecniche proprie dei database come indexing
- **Set matching**
  - Matching di un alto numero di risorse usando una espressione compatta
- **Indirect references**
  - Per permettere ad un ClassAd di far riferimento ad un altro