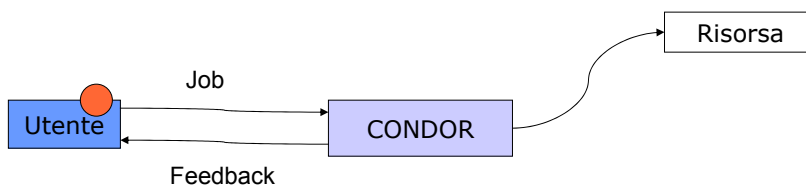


Condor e la Griglia

Jaskaran Singh
CS-599 Introduction to Grid
Computing

Cosa Fa Condor



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

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

Condor tutorial NCSA Alliance

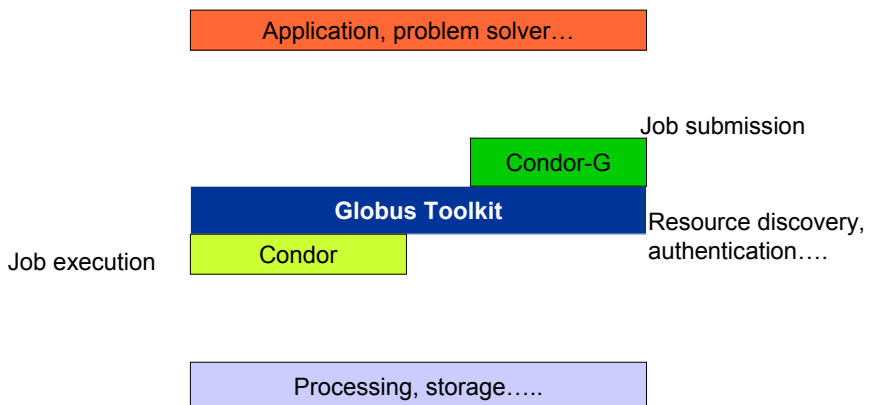
Condor: Un Sistema per High Throughput Computing

- 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

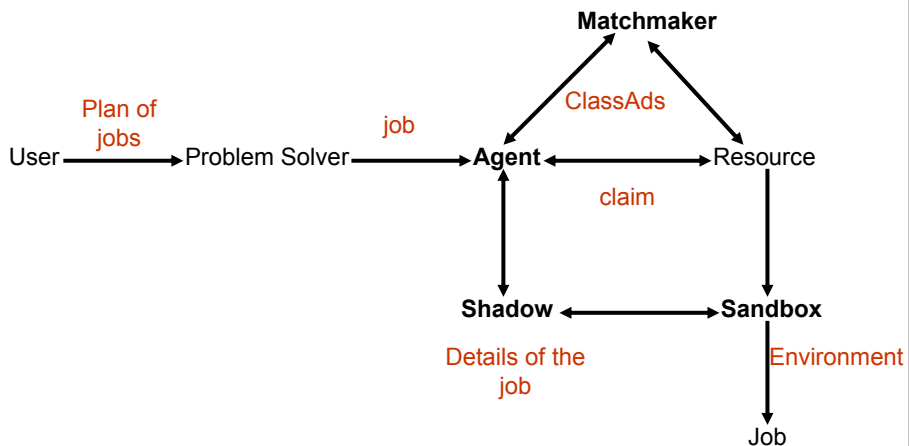
Condor-G: Agente di gestione Computazionale per Grid Computing

- Combinazione di tecnologia Globus e Condor
- Globus
 - ◆ Protocolli per comunicazioni sicure tra domini
 - ◆ Accesso standard a sistemi batch remoti
- Condor
 - ◆ Job submission e allocation
 - ◆ Error recovery
 - ◆ Creazione di un ambiente di esecuzione

Tecnologie Condor in Middleware di Grid



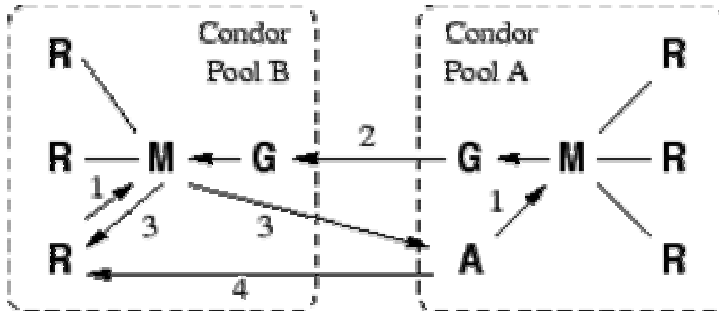
Condor Kernel



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

Gateway Flocking



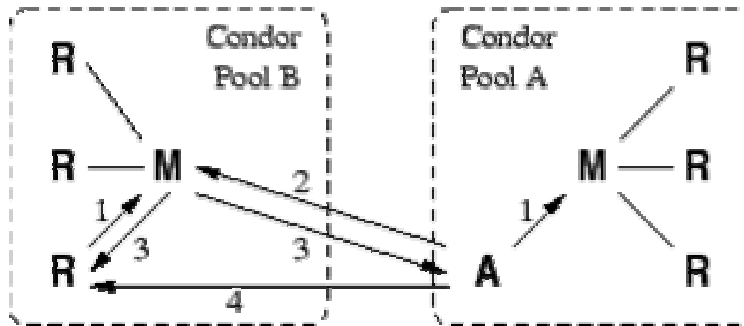
I Gateway passano informazione sui partecipanti tra pool, MA invia richiesta a MB attraverso un gateway G, MB ritorna un *match*

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

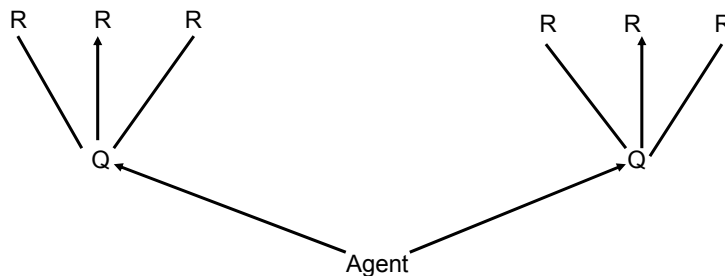
Direct Flocking



A interagisce anche con il Condor Pool B

Condor-G

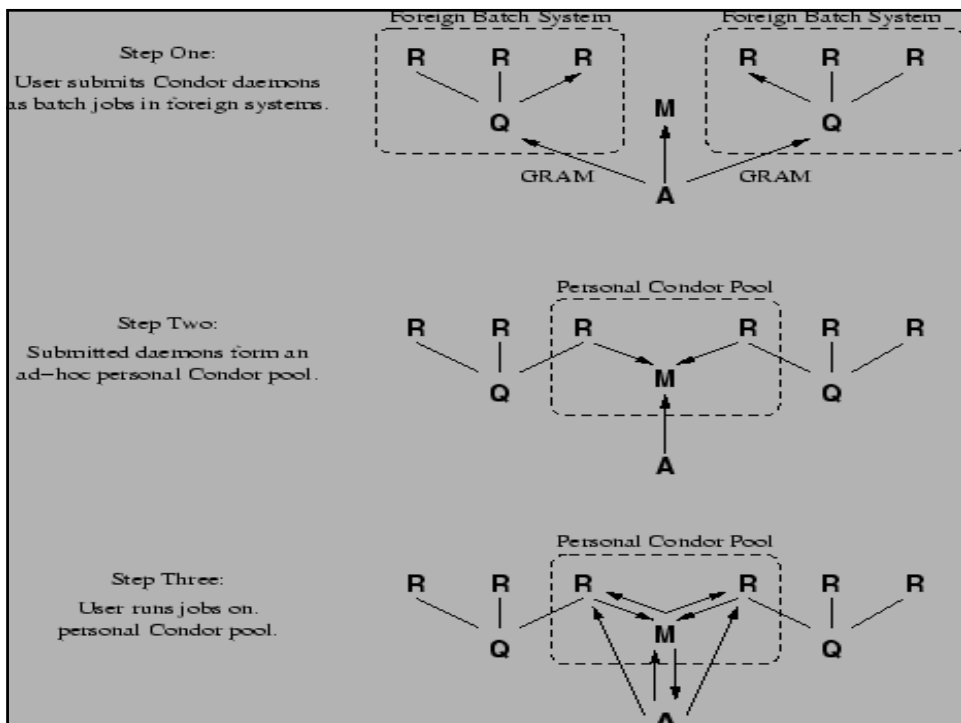
- Agenti Condor che comunicano con il GRAM



Jobs vengono accodati e non direttamente assegnati alle risorse

Direct Flocking Vs Condor-G

- In Condor un Agente sottomette job ad una risorsa, in Condor-G il job è sottomesso ad una coda.
- Agenti in Condor-G possono
- **Over subscribe**
Sottomettere un job a più code, attendere una risposta e quindi cancellare gli altri job
 - **Under subscribe**
Sottomettere un job a una coda che può essere lunga
- Il GRAM permette l'accesso ad una varietà di sistemi batch, con accodamento ed esecuzione remota



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
- Autorizzazione e autenticazione indipendenti dalle richieste

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 = "CompSci"  
Owner = "tannenba"  
DiskUsage = 6000  
]
```

Machine ClassAd

```
[  
MyType="Machine"  
TargetType="Job"  
Machine="tnt.isi.edu"  
Requirements=  
(Load<3000)  
Rank=dept==self.dept  
Arch="Intel"  
OpSys="Linux"  
Disk=600000  
]
```

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

Planning e Scheduling

- Planning
 - ◆ Acquisizione di risorse da parte degli utenti
 - ◆ E' relativo a 'cosa' e 'dove'
- Scheduling
 - ◆ Gestione di una risorsa da parte del proprietario
 - ◆ E' relativo a 'chi' e 'quando'
- Feedback tra planning e scheduling

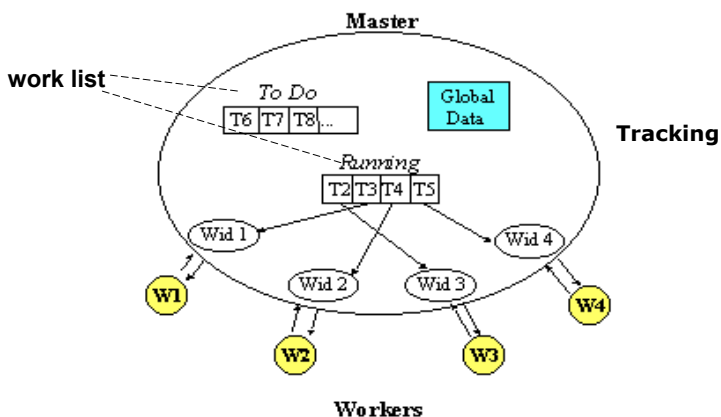
Planning e Scheduling

- Pianificare sulla base di uno schedule
 - ◆ Lo scheduler pubblica informazioni circa orari e priorità
 - ◆ Condor-G pianifica in questo modo
- Scheduling senza una pianificazione
 - ◆ Match dei risultati di una richiesta di risorsa
 - ◆ Un agente crea uno schedule per eseguire i task su quella risorsa

Problem Solver

- Struttura di alto livello costruito 'sopra' un Condor agent
- Si occupa dell'ordinamento dei job e della selezione dei task
- E' esso stesso rappresentato come un job
 - ◆ Un job che sottomete jobs
- Dipende dall'agent per l'esecuzione dei task
- *Master-Worker and DAG Manager*

Master-Worker

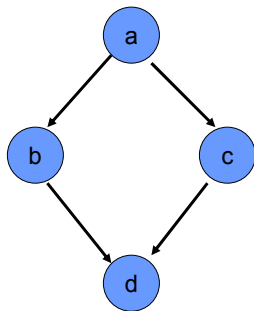


T2 in esecuzione su W3 e monitorato da Wid 3

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

- 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