ESD Electronic Systems Design

Synthesis Verification Testing Power Communication

Gestione dei processi in sistemi distribuiti

Sommario

- Allocazione e migrazione dei processi
- Scheduling in sistemi distribuiti

Allocazione e migrazione dei processi

- In un sistema distribuito è possibile distribuire il carico su processori o host diversi
- Motivazioni
  - Bilanciamento del carico (load balancing)
    - Tutti i processori eseguono un carico simile
  - Utilizzo di risorse speciali
    - Processi che utilizzano risorse non disponibili ovunque
  - Ottimizzazione delle comunicazioni
    - Allocazione di processi strettamente connessi o utilizzo di “tanti” dati remoti
  - Sopravvivenza a crash preventivati
    - Processi con lunghe esecuzioni possono migrare su altre macchine
- Particolarmente adatto a Cluster

Allocazione vs. migrazione

- Tecnicamente migrazione = caso particolare di allocazione
- Convenzionalmente
  - Allocazione = allocazione statica (non migratoria)
    - Decisione sull'allocazione di un processo fatta all'atto della sua creazione
    - Una volta creato, un processo non può essere spostato
    - Non-preemptive
  - Migrazione = allocazione dinamica
    - Un processo può essere spostato in qualunque momento della sua esecuzione
    - Preemptive

ALLOCAZIONE STATICA

Allocazione: problematiche

- Problematiche di progetto
  - Tipo di decisione (= algoritmo)
    - Centralizzato vs. distribuito
    - Locale vs. globale
      - Il processo migra dopo un'analisi del carico locale o dopo un'analisi globale dello stato di tutti gli host?
    - Iniziato da mittente vs. iniziato da destinatario
      - Inizia chi è sovraccarico o chi è "annoiato"?
    - Deterministico vs. non deterministico
    - Ottimo vs. sub-ottimo

Allocazione: problematiche

- Problematiche di implementazione
  - Ogni macchina deve conoscere il proprio carico
    - Come si misura?
      - N° processi in esecuzione/attesa
      - Utilizzo CPU
  - Valutazione e analisi dell'overhead
    - Quanto costa allocazione remota?
  - Complessità degli algoritmi
    - Meglio semplice e sub-ottimo o complesso e ottimo?
  - Che cosa trasferire di un processo?

Algoritmi di allocazione

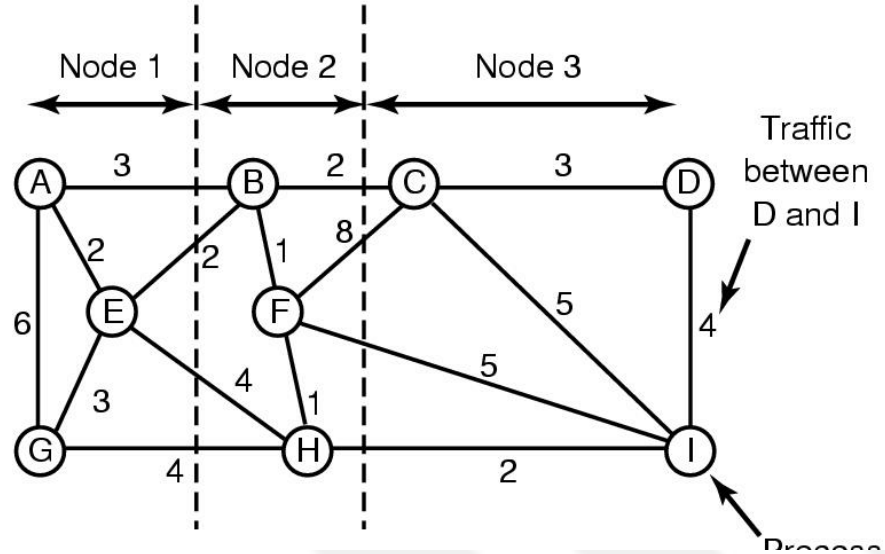
- Algoritmi deterministici
- Algoritmi centralizzati
- Algoritmi distribuiti

Algoritmi di allocazione deterministici

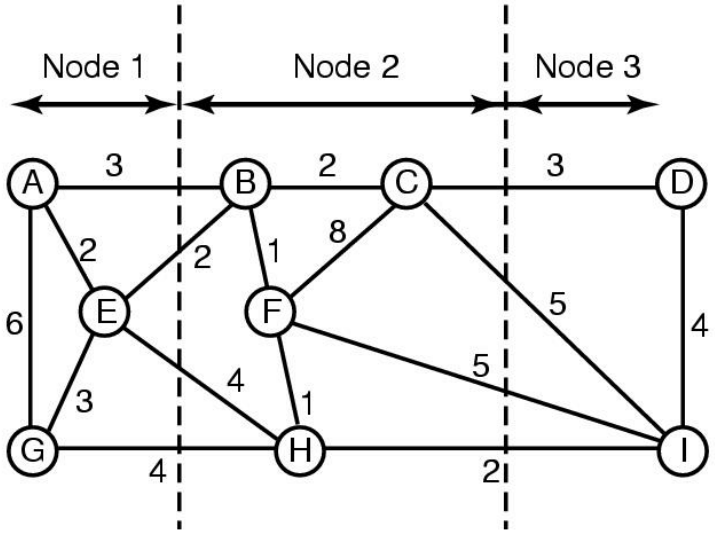
- Determinismo richiede conoscenza a priori dei requisiti di CPU e memoria
  - Raramente disponibile → approssimazioni
- Modello del sistema: grafo
  - Nodi = N processi
  - Archi = flusso di messaggi
  - Peso degli archi = “traffico” tra nodi
- Allocazione basata su algoritmi di teoria dei grafi
  - Rispettare i vincoli (memoria, CPU, …)
  - Minimizzare il traffico sulla rete

Algoritmi di allocazione deterministici

- Esempio



Traffico totale sulla rete = 30



Traffico totale sulla rete = 28

Algoritmi di allocazione centralizzati (UP-DOWN)

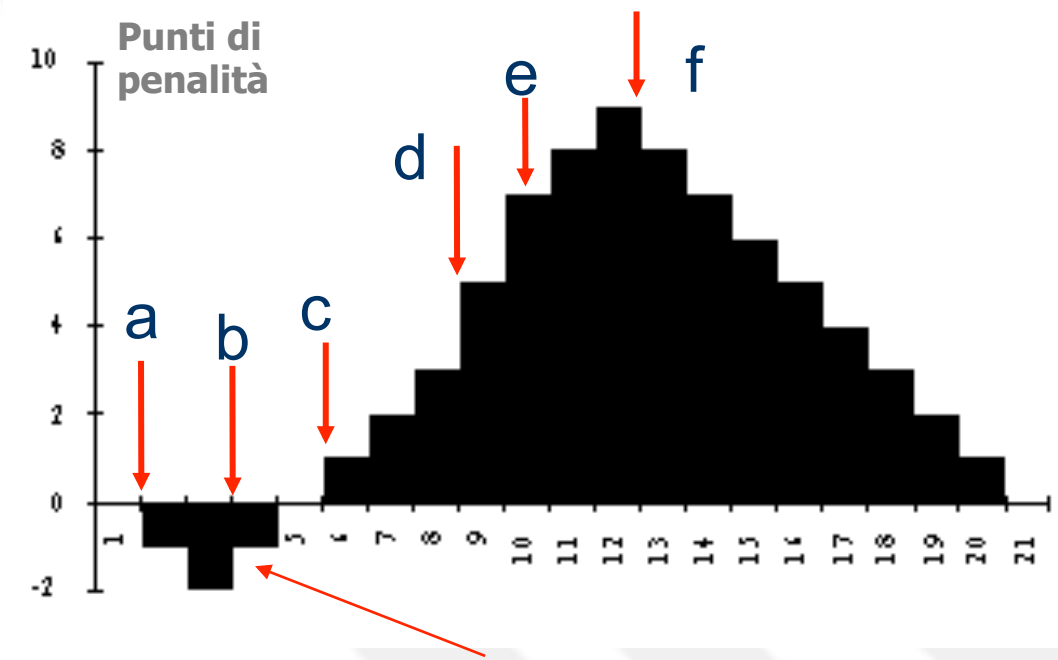
- [Mutka&Livny 87]
- Un processo coordinatore mantiene una tabella di utilizzo con una entry U(w) per ogni host w del sistema
  - Inizialmente U(w)=0
- Per ogni evento significativo, U viene aggiornata
  - Quando w “esegue processi su un host remoto”, U(w) accumula punti di penalità (da sommare alla sua entry)
    - N punti al secondo
  - Se w ha richieste pendenti (processi da allocare), viene messo in attesa, U(w) perde punti di penalità (da sottrarre alla sua entry)
  - Se w non ha richieste pendenti e non “usa” altri processori, U(w) viene lentamente portato verso 0

Algoritmi di allocazione centralizzati (UP-DOWN)

- Quando si libera un host
  - Coordinatore sceglie per l'allocazione una richiesta pendente dell'host con il punteggio più basso
- Basato sull'allocazione "democratica" dei processori

Algoritmi di allocazione centralizzati (UP-DOWN)

- Esempio



a) Arriva P1
b) P1 allocato su altro processore
c) Arriva P2
d) P2 allocato su altro processore
e) P1 termina
f) P2 termina

Attende prima che P1 venga allocato

Algoritmi di allocazione distribuiti

- Un host decide per l'allocazione remota di un nuovo processo in base al suo carico
- Come sceglie l'eventuale host remoto?
  - Scelta casuale di una macchina a catena
    - Processo inviato a quella macchina
    - Se sovraccarica, questa sceglie nuovamente a caso
    - max n° di trasferimenti fissato
  - Scelta casuale di una macchina
    - Richiesta di accettazione per verifica del carico
    - Se rifiutata, richiesta iterata ad altri fino ad un certo numero di volte
  - Scelta casuale di n macchine
    - Processo inviato alla macchina meno carica

Algoritmi di allocazione distribuiti

- Analisi degli algoritmi distribuiti [Eager 86]
  - Algoritmo 3 meglio degli altri, ma di poco
  - Algoritmo 2 più semplice

ALLOCAZIONE DINAMICA

Migrazione di processi

- Problematiche [Milojcic&al 96]
  - Chi inizia la migrazione?
  - Quale porzione del processo viene migrata?
  - Che cosa succede ai segnali di I/O pendenti quando un processo viene migrato?

Chi inizia la migrazione?

- Il processo stesso
  - Se l'obiettivo è l'accesso a speciali risorse remote
  - Non trasparente al processo
- Il "sistema"
  - Se l'obiettivo è il bilanciamento del carico
  - Trasparente al processo
    - I sistemi operativi mittente e destinatario si fanno carico della migrazione

Cosa migra di un processo?

- Process Control Block (facile)
- Spazio di indirizzamento
- Elenco dei file aperti

Trasferimento dello spazio di indirizzamento

- Assunzione: memoria virtuale con paginazione/ segmentazione
- Eager (all)
  - Trasferimento dell'intero spazio di indirizzamento
  - Può essere inutilmente costoso
- Pre-copy
  - Trasferimento avviene mentre il processo continua ad eseguire sul nodo originale
  - Eventuali modifiche in questo periodo vanno riscritte successivamente

Trasferimento dello spazio di indirizzamento

- Eager (dirty)
  - Trasferimento solo delle pagine in memoria modificate
  - Eventuali altre pagine verranno trasferite su richiesta
  - Il nodo originale resta coinvolto
- Copy-on-reference
  - On-demand puro

Trasferimento file aperti

- Considerazioni analoghe al trasferimento dello spazio di indirizzamento
  - File migra se è aperto solo dal processo che migra
  - File migra solo alla prima richiesta di accesso
    - Se il processo non usa il file durante la sua esecuzione remota, il file non migra

Segnali di I/O pendenti

- Segnali e messaggi “in sospeso” vengono memorizzati temporaneamente sull’host di origine e inviati successivamente verso il nuovo host

Schemi di migrazione – Esempi

- Charlotte [Artsy&Finkel 89]
- Condor [Wisconsin]

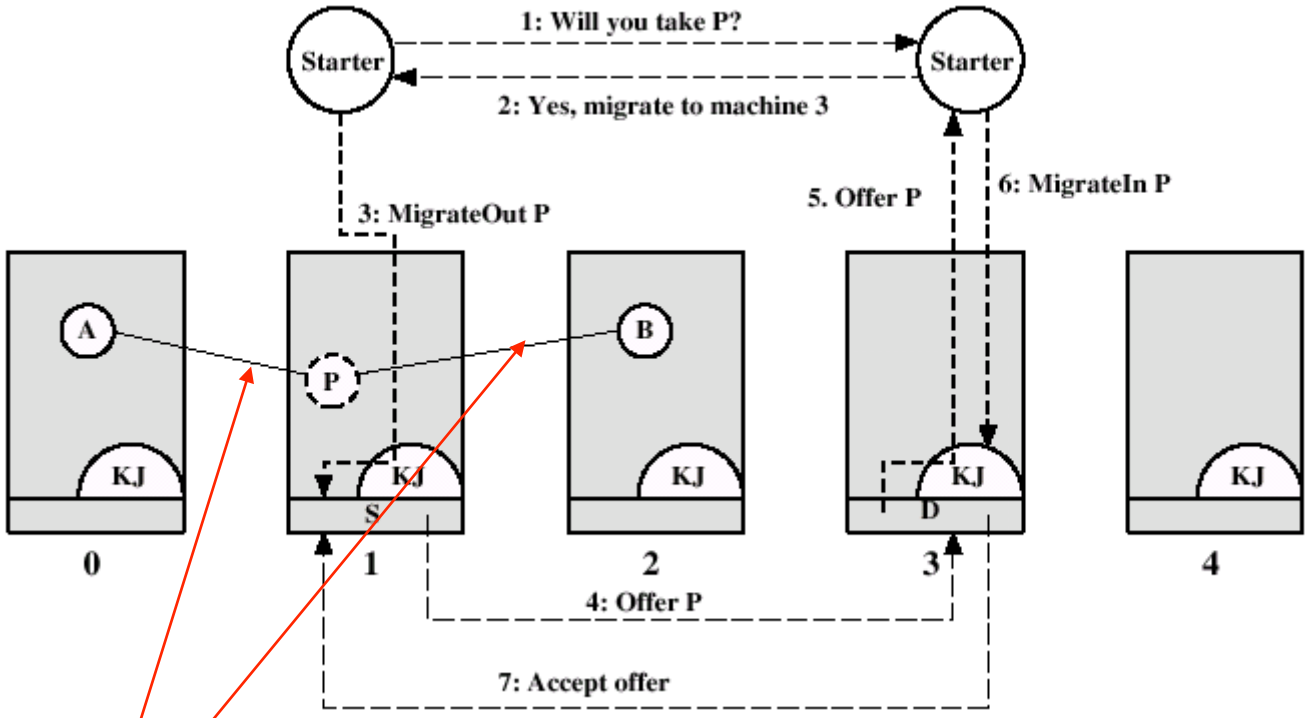
Charlotte

- Politica di migrazione (chi, quando e dove) è responsabilità dello *Starter*
  - Programma di sistema (eventualmente 1 per macchina)
  - Riceve statistiche sull'utilizzo delle varie macchine

Charlotte

- Negoziazione tra due Starter (SS e SD)
  - SS decide che P (locale su S) deve essere migrato su D, e lo comunica a SD
  - Se SD può accettare ➡ acknowledge
  - SS comunica al kernel di S la decisione
  - Kernel di S offre P al kernel di D, insieme a statistiche
  - D può rifiutare. Se accetta, passa l'offerta a SD
  - La decisione finale di SD viene comunicata a D
  - D prenota le risorse necessarie e invia l'accettazione a S

Charlotte



Nota: la procedura deve prevedere anche l'aggiornamento di eventuali canali di comunicazioni aperti

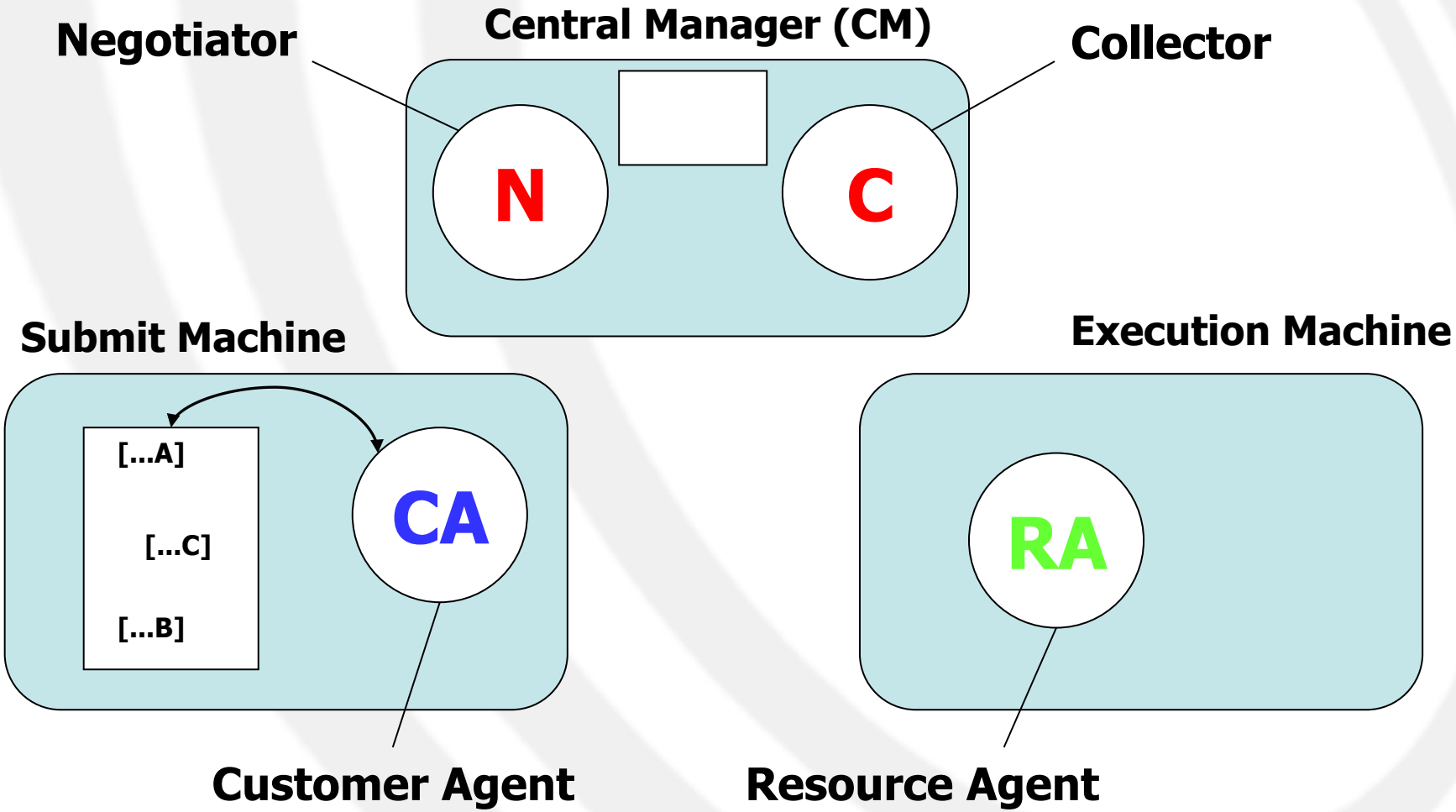
Condor

- Sviluppato alla University of Wisconsin
- Sistema per l'utilizzo di CPU inattive per il calcolo distribuito
- Utilizzabile anche su WAN
  - Registrazione della propria macchina in un Condor Pool
  - >300 Pool nel mondo con oltre 8500 CPU
- http://www.cs.wisc.edu/condor/

Condor

- Completamente (quasi) trasparente all'utente
  - Installazione sulla propria macchina di un software opportuno per esecuzione remota (in ingresso ed in uscita)
  - No account su macchine remote
  - Compilando le applicazioni con la libreria Condor
    - System call intercettate dalla libreria su host remoto
    - Condor invia le system call dalla macchina remota a quella di partenza (tipo RPC)
      - Es.: per letture da disco
  - Dati restano sulla macchina di partenza

Condor – Struttura



Condor – Customer Agent

- Si trova nella macchina che sottomette il job
- Mantiene una coda di job eseguiti
- Comunica il suo stato al CM
- Sceglie i job da eseguire

Condor – Resource Agent

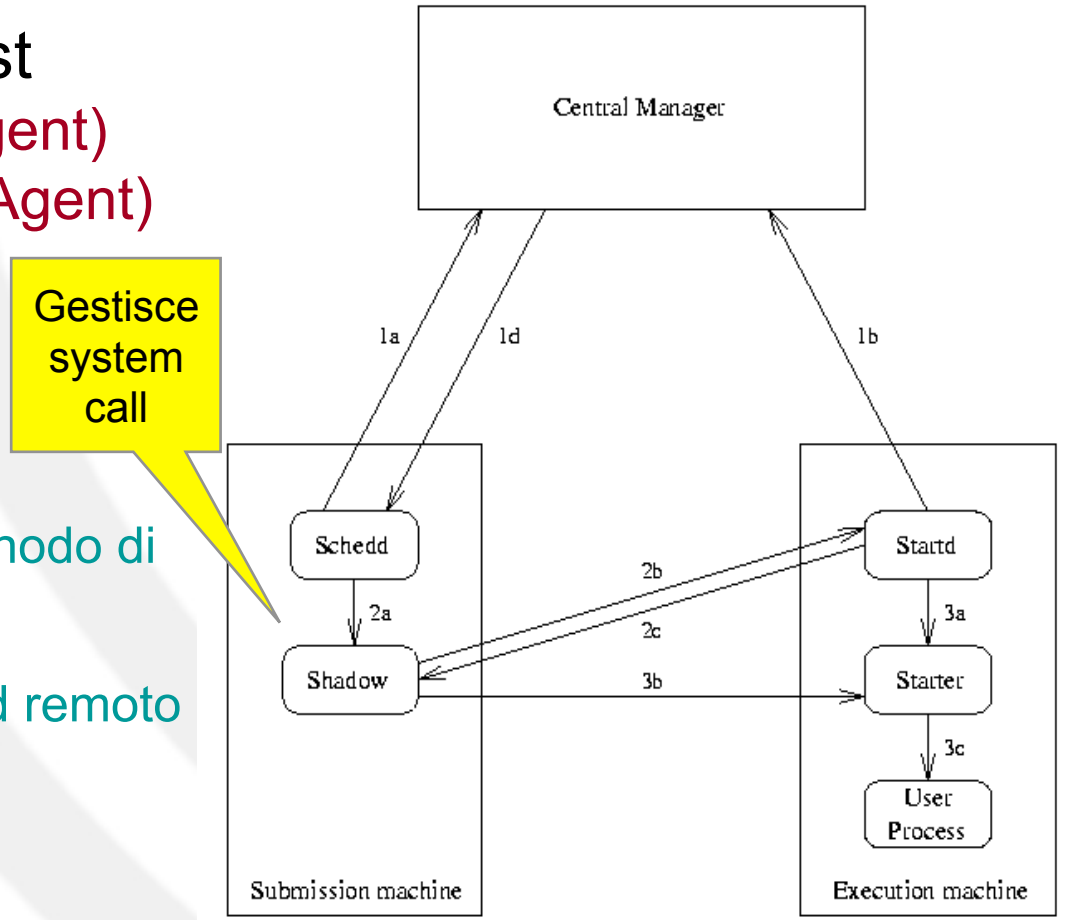
- Monitora lo stato del sistema
  - Carico medio
  - Inattività di mouse e tastiera
  - Spazio su disco, memoria, …
- Comunica lo stato al CM
- Attende richieste per l'esecuzione di job

Condor – Central Manager

- Collector
  - Accetta informazioni da resource e customer agent
- Negotiator
  - Cerca di “accoppiare” customer e resource
- Accountant
  - Registra utilizzo delle risorse da parte dei customer

Condor – Protocollo

- Due daemon per host
  - Startd (Resource Agent)
  - Schedd (Customer Agent)
- 3 Fasi
  - Matchmaking
    - CM raccoglie info
  - Connection
    - Shadow creata sul nodo di partenza
    - Negozia l'avvio del processo con Startd remoto
  - Execution



SCHEDULING IN SISTEMI DISTRIBUITI

Scheduling in sistemi distribuiti

- Non molto diverso da caso non distribuito!
  - Ogni macchina ha il suo scheduler, che opera indipendentemente dagli altri
- Non efficiente nel caso di processi remoti strettamente interagenti
- Es.:
  - Scheduling locali = RR
  - A,D e B,C comunicano pesantemente tra loro

	proc	
Time slot	0	1
0	A	C
1	B	D
2	A	C
3	B	D
4	A	C
5	B	D

Co-scheduling

- Scheduling che tiene conto dei pattern di IPC
  - Definisce gruppi di processi altamente “connessi”
  - Garantisce che i processi appartenenti ad un gruppo siano in esecuzione nello stesso slot
- Esempio

	Processors							
Time slot	0	1	2	3	4	5	6	7
0	X				X			
1			X			X		
2		X			X		X	
3	X					X		
4		X		X				X
5			X		X			

- Processi in ogni slot eseguono simultaneamente su ogni processore
- Round-robin locale, slot di ampiezza fissa (uguale per tutti)
- Context switch e sincronizzazione dei time slice via broadcast