Párhuzamos és Grid rendszerek (14. ea)

Összefoglalás

Szeberényi Imre
BME IIT

<szebi@iit.bme.hu>

Összefogalás

- Párhuzamos architektúrák
- Párhuzamos programok modellezése
- Párh. prog. fejlesztési módszerek
  - kevés algoritmus
- Fejlesztési környezetek, nyelvek
  - PVM, MPI
  - OpenMp
  - Clearspeed
  - CUDA

Jellemző szupersz.gép típusok

- Vektorprocesszoros rendszerek
  - Gyors műveletvégzés vektor jellegű adatokon
- Masszívan párhuzamos rendszerek (MPP)
  - Üzenetküldéses elosztott memóriás (MDM)
  - Szimmetrikus multiprocesszoros (SMP)
  - Elosztott közös memória (DSM)
- Elosztott számítási rendszerek
  - Homogén rendszerek
  - Heterogén rendszerek
- Metaszámítógépek és Grid redszerek

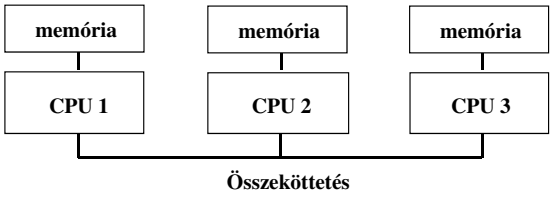
Párhuzamos gépek osztályai

- Szimmetrikus multiprocesszoros (SMP)
  - sok azonos processzor közös memóriával
  - egy operációs rendszerrel
  - NUMA, ccNUMA
- Masszívan párhuzamos (MPP)
  - sok processzor gyors belső hálózattal
  - elosztott memória
  - sok példányban fut az operációs rendszer
- Klaszter
  - sok gép gyors hálózattal összekötve
  - elosztott memória
  - sok példányban esetleg heterogén operációs rendszer

Flynn-féle architektúra modell

INSTRUCTIONS \ DATA	Single	Multiple
Single	**S**ingle **I**nstruction **S**ingle **D**ata **SISD** (serial machines)	**S**ingle **I**nstruction **M**ultiple **D**ata **SIMD** (vector processors)
Multiple	**M**ultiple **I**nstruction **S**ingle **D**ata **MISD** (pipelines)	**M**ultiple **I**nstruction **M**ultiple **D**ata **MIMD** (multiprocesszors)

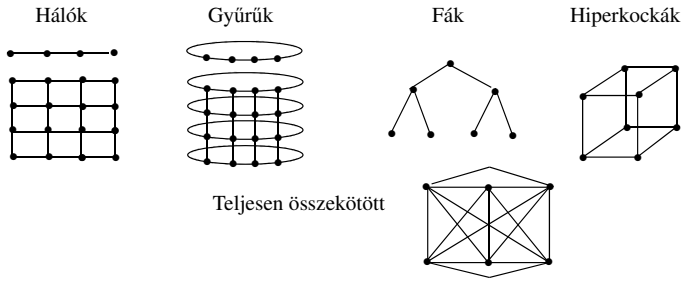
Idealizált párhuzamos számítógép



- Több processzor egyazon problémán dolgozik.
- Minden processzornak saját memóriája és címtartománya van.
- Üzenetekkel koordinálnak és adatokat is tudnak átadni.
- A lokális memória elérése gyorsabb.
- Az átviteli sebesség független a csatorna forgalmától.

Architektúrák jellemzői

- Processzorok eloszlása
- Homogén vagy heterogén
- A kapcsolat késleltetése és sávszélessége
- Topológia



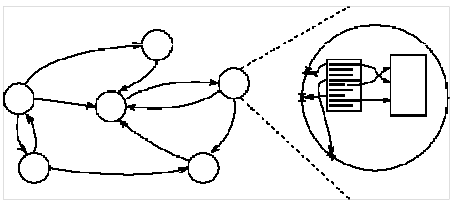
Programozási modell

- Közös memóriás
- Elosztott közös memóriás
- Üzenet küldéses

Valójában egyik modell sem kötődik szorosan a tényleges architektúrához

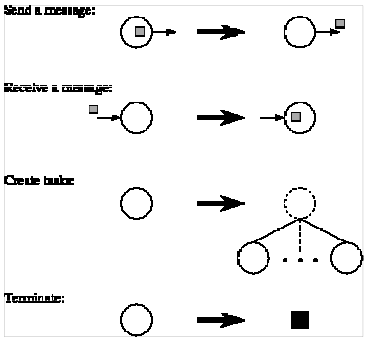
Taszk/csatorna modell /1

- minden taszk szekvenciális programot futtat
- minden taszknak van saját memóriája
- taszkok csatornákkal kapcsolódnak
- a csatornák üzenetsorokat valósítanak meg



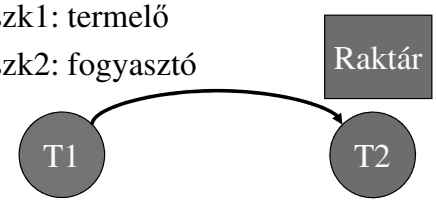
Taszk/csatorna modell /2

- taszkok konkurensek
- van lokális memóriájuk
- küldés aszinkron
- fogadás szinkron
- csatornához in/out portokkal csatlakoznak
- taszkok tetszőlegesen rendelhetők össze a processzorokkal



Taszk/csatorna modell /3

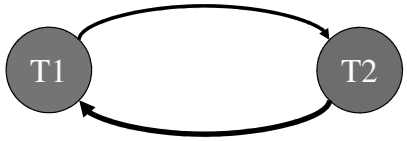
- Példa: termelő-fogyasztó probléma
  - taszk1: termelő
  - taszk2: fogyasztó



- ha a fogyasztó lassabb, akkor a felhalmozódik a termelt termék
- ha a termelő a lassabb, a akkor vár a fogy.

Taszk/csatorna modell /4

- Példa: termelő-fogyasztó probléma
  - taszk1: termelő
  - taszk2: fogyasztó



- második csatornán a fogyasztó jelzi, ha kér újabb terméket
- a termelő ennek hatására termel

Taszk/csat. modell jellemzői

- A modell közvetlenül hozzárendelhető az idealizált számítógéphez.
- A taszk egy soros kódot reprezentál.
- A csatorna processzorok közötti kommunikációt valósít meg.
- A taszk működése független a taszk-processzor összerendeléstől, taszkok számától.
- Moduláris felépítést tesz lehetővé.

Taszk/csatorna vs. üzenet

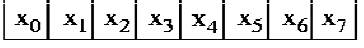
- Az üzenet egy adott taszknak szól, ezért kevésbé absztrakt, mint a csatorna.
- Az általános üzenetküldéses modell szerint nem lehet dinamikusan új taszkot létrehozni. (Több megvalósításban lehet.)
- Egy processzor csak egy taszkot futtathat. (Több megvalósításban ez sem korlát.)

Párh. algoritmus példák /1

- Véges differenciák:
  - egy vektor minden elemére T-szer végre kell hajtani a következő műveletet:

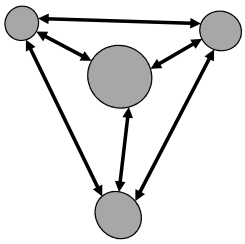
$$0 < i < N-1,\ 0 \le t < T : X_i^{(t+1)} = \frac{X_{i-1}^{(t)} + 2X_i^{(t)} + X_{i+1}^{(t)}}{4}$$

  - Minden elemet egy-egy taszk számol, aki kommunikál a szomszédaival:



Párh. algoritmus példák /2

- Páronkénti iteráció (pl. atomok kölcsönös egymásra hatása)

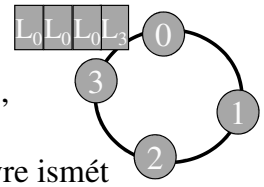


$$f_i = \sum_{j=0}^{N-1} F(X_i, X_j)$$

  - N*(N-1) üzenet kell, esetleg N*(N-1)/2, ha kihasználjuk a szimmetriát.

Párh. algoritmus példák /3

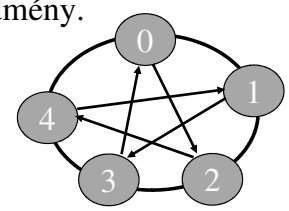
- Körkörös kapcsolat (csatorna) a fenti problémára hatékonyabb üzenetstruktúrát eredményez:
  - Egy N elemű vektorba minden taszk beteszi a saját adatát (koord., tömeg) és elküldi a szomszédnak.
  - A bejövő üzenetbe megfelelő helyre ismét elhelyezi a saját adatát és továbbküldi azt.
  - N-1 lépés után mindenki ismeri az a többiek koordinátáit és tömegét.
  - F értéke minden lépésben az új partnerek adata alapján akkumulálható.



Párh. algoritmus példák /4

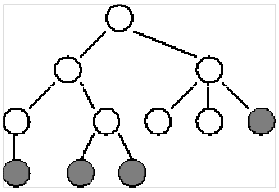
- N újabb csatornával az algoritmus a szimmetria miatt tovább egyszerűsíthető:
  - hozzunk létre minden i. taszk és i+N/2-dik taszk között egy újabb csatornát.
  - az adott atomra ható erőket folyamatosan számoljuk, és küldjük is körbe.
  - N/2 iterációval előáll az eredmény.

L_0	L_1	L_2	L_3	L_4
F_0	F_1	F_2	F_3	F_4

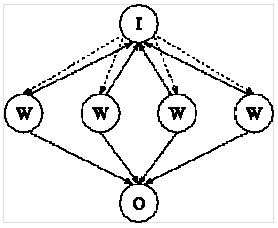


Párh. algoritmus példák /5

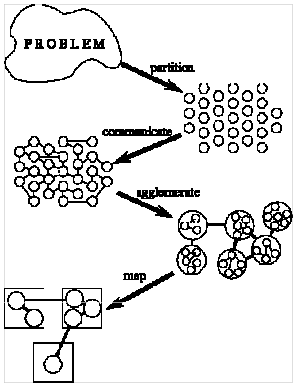
- Párhuzamos keresés:
  - fában történő keresés egyszerűen párhuzamosítható



- Paraméter elemzés:
  - master-worker algoritmus



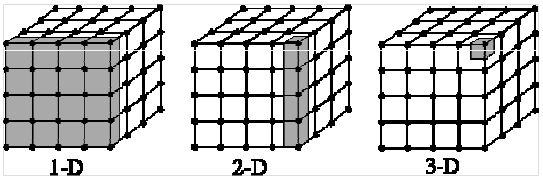
PCAM módszertan



1. Particionálás: Részfeladatokra osztás. NEM veszi figyelembe a fizikai gép adottságait.
2. Kommunikáció megtervezése: Részfeladatok közötti adatcsere és szinkronizációs séma kialakítása.
3. Agglomeráció: Részfeladatok nagyobb egységekbe gyűjtése a hatékonyságnövelés érdekében.
4. Leképezés: A részfeladatok processzorhoz (feldolgozó elemhez) rendelése.

Domén dekompozíció

- Adat vagy paramétertér felosztása. Az adat lehet input, output, vagy közbülső adat.
  - Példa: Egy 3D rácson minden rácspontban ki kell számolni egy értéket. 1, 2, vagy 3 dimenziós partíció:



Funkcionális dekompozíció

- Az algoritmus felosztása olyan részekre, melyek párhuzamosíthatók.
- Alapvetően a feladat funkcióiból adódik.
- Az adatokra is figyelni kell.
- Tipikus példa, amikor az adatok partícionálása nem járható: keresés fában.
  - funkcionálisan viszont bontható

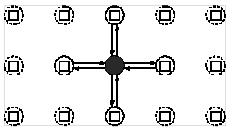
Kommunikáció

- Kis környezetű (local) és globális
  - a taszkok csak kis környezetükben (szomszéd), vagy sok másik taszkkal is kommunikálnak.
- Strukturált és nem strukturált
  - rács, gyűrű, ... vagy más
- Statikus és dinamikus
  - végrehajtás közben változik
- Szinkron vagy aszinkron
  - koordináció hiánya

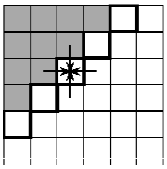
Kommunikációs példák /1

Lokális kommunkáció (véges elem):

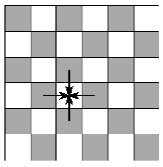
$$X_{i,j}^{(t+1)} = \frac{4X_{i,j}^{(t)} + X_{i-1,j}^{(t)} + X_{i+1,j}^{(t)} + X_{i,j-1}^{(t)} + X_{i,j+1}^{(t)}}{8},$$



$$X_{i,j}^{(t+1)} = \frac{4X_{i,j}^{(t)} + X_{i-1,j}^{(t+1)} + X_{i+1,j}^{(t)} + X_{i,j-1}^{(t+1)} + X_{i,j+1}^{(t)}}{8},$$

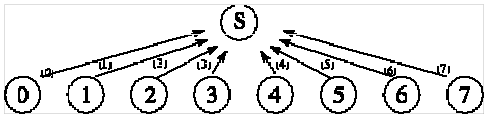


Red-Black ordering:

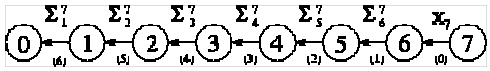


Kommunikációs példák /2

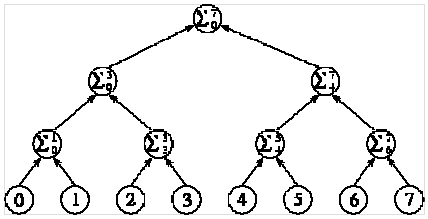
Globális kommunkáció (szumma):



Csővezeték:

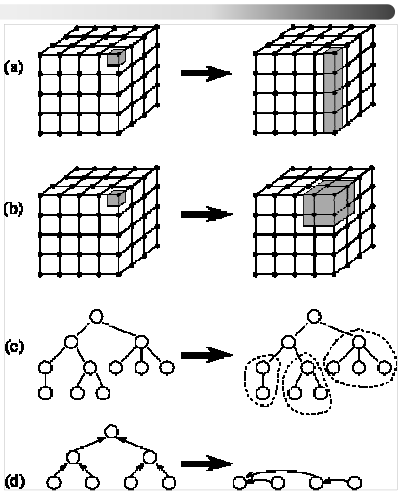


Oszd meg és uralkodj:



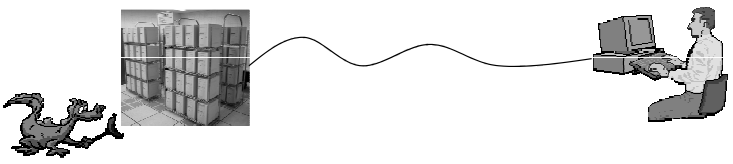
Agglomeráció

- A tényleges párhuzamos gép kommuniká-ciós adottságait is figyelembe véve a részfeladatokat nagyobb egységekbe gyűjtjük.



Cluster koncepció

- Gyors hálózattal összekapcsolt gépek
- Gyakran közös fájlrendszer
- CPU vagy tárolási kapacitás növelése
- Paraméter study, vagy párhuzamos alkalmazások



Ütemezők

- Condor (University of Wisconsin)
- DQS (Florida State University)
- LoadLeveler (IBM)
- Maui, Moab (Cluster Resources)
- LSF (Platform)
- PBS, OpenPBS (Alatair)
- Sun Grid Engne (SUN)
- Torque (Cluster Resources)

Elosztott fájlrendszerek

- Nagyméretű klaszterekhez
- Földrajzilag is elosztott rendszerekhez
  - NFS
  - AFS, CODA
  - Lustre, SFS
  - GFS
  - GlusterFS
  - OCFS
  - Gfarm file system
  - Google file system
  - GPFS
  - BigTable
  - Parallel Virtual File System
  - QFS

Több mint 70! fs

AFS (Andrew File System)

- Elosztott fájlrendszer, ami fájlok megosztására alkalmas lokális és távolsági hálózaton.
- Transzparens fájlhozzáférést biztosít.
- Az NFS-hez hasonló, annak alternatívájaként jött létre.
- Ma az OpenAFS számos UNIX, LINUX, WinX platformon elérhető.

AFS alapfogalmai

- Cellák
- Kötetek
- Tokenek
- Cache menedzser
- Fájl védelem
- Fájl névtér

Lustre architektúra

- Három fő funkcionális egysége van:
- Metadata szerver (MDS), ami a fájl neveket, katalógusokat, védelmi kódokat és egyéb metaadatot tárol.
- Object storage szerverek (OSS), melyek az adatokat tárolják.
- Kliens ami az adatokat felhasználja, létrehozza.

Lustre architektúra /2

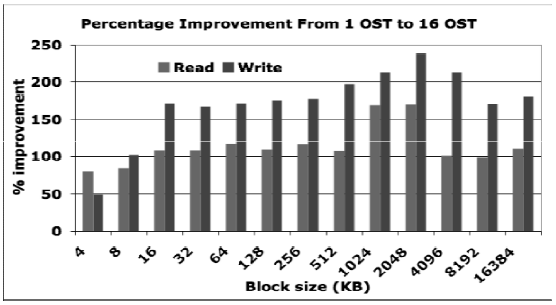
- Az adatok logikai kötetmenedzsmenttel ellátott RAID tárolókban tárolódnak, amit az OSS és az MDS dedikált módon használ.
- Jelenleg egy módosított ext4 fájlrendszer a logikai tároló. ZFS support (béta)
- Amikor egy kliens fájlt akar elérni, először az MDS-ben meg kell keresnie.

Lustre architektúra /2

- A fájl egyes darabjai több OSS-en tárolódhatnak, ami a kliens és az OSS között szűk keresztmetszet kialakulását gátolja.
- A kliensek nem módosítják közvetlenül az OSS-ben tárolt adatokat, hanem ezt a OSS-re bízzák, szemben a GFS megoldásával.
- Ez a módszer növeli a megbízhatóságot és a hibatűrést.

Skálázhatóság teljesítmény

- TOP 500-as lista tetején (Titan is)
- Skálázhatóság, nagy rendelkezésre állás
- Üzleti szupport (Oaracle-n kívül mindenki)



S. Saini, J. Rappleye, J. Chang, D. Barker, P. Mehrotra, R. Biswas: I/O Performance Characterization of Lustre and NASA Applications on Pleiades

OpenMP motiváció

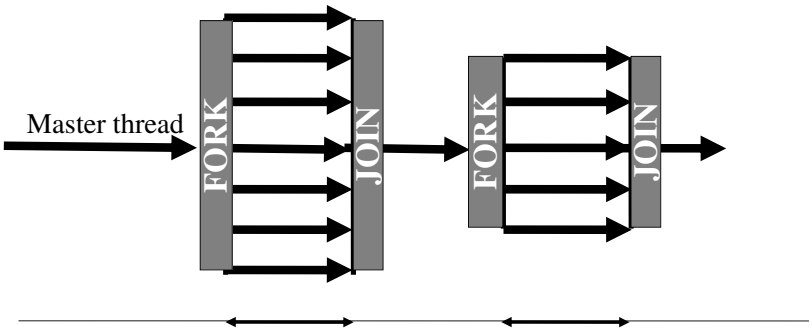
- Szálakkal történő párhuzamosítás macerás:
- OS függő API-k:
  - Windows: CreateThread
  - UNIX: pthread_create
- Még egy vektor elemeinek összeadásához is sok ismeret kell:
  - Kölcsönös kizárás (mutex)
  - Külön soros és párhuzamos kód keletkezik
  - Nehéz a skálázhatóság megoldása

OpenMP

- Nyelvi kiterjesztés
- A programozó a funkcionalitásra koncentrálhat.
- A párhuzamosítás csak lehetőség.
- Shared memóriás párhuzamosítás
- Ipari szabvány
- 1997: 1.0
- 2011: 3.1
- 2013: 4.0 – jelenleg draft

http://openmp.org/mp-documents/OpenMP3.1-CCard.pdf

Végrehajtási modell



Shared memoria modell

- A szálak változókon keresztül kommunikálnak.
- A megosztás nyelvi szinten definiált
- Versenyhelyzet kialakulhat
  - Szinkronizációs eszközök
  - Megosztás minimalizálása

Szintaxis

- #pragma omp construct [clause [clause] …]
- Egy blokkra vonatkozik (egy belépés, egy kilépés)
- OpenMP konstrukciók:
  - Parallel régiók megadása
  - Munka elosztás (work sharing)
  - Adatelérés szabályozása
  - Szinkronizáció
  - Runtime függvények

Általános célú GPU

- A programozható vertex és fragment shaderek beépítésével általános célú eszközzé vált.
- Vektorprocesszor (SIMD), de pipeline egységek is vannak benne (MISD).
- Jellemzően SIMD
- Programozás: CUDA, OpenCl, Cg, …

Egy példa

NVIDIA Quadro FX5800 grafikus kártya

- PCIe x16
- 240 CUDA mag
- 4 GB DDR3
- 78 GFlos double
- 933 Gflops single
- 189 W
- 300Millió háromszög / sec
- NVIDIA CUDA

PRAM modell

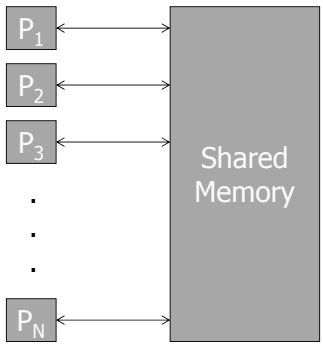
- Parallel Random Access Machine (PRAM)
- Elméleti modell a az algoritmusok vizsgálatához

Célja:

- Algoritmusok osztályozása, komplexitásának vizsgálata.
- Párhuzamosíthatóság elvi határainak felfedése.
- Új algoritmusok kifejlesztése.

PRAM modell /2

- Végtelen memória és processzorszám
- Nincs direkt kommunikáció a processzorok között:
  - csak a memóriában kommunikálhatnak
  - aszinkron működésűek
- A processzorok tetszőlegesen hozzáférnek a memóriához.
- Hozzáférés 1 ciklus
- Tipikusan minden processzor ugyanazt az algoritmust hajtja végre. (read, compute, write)



Memória hozzáférés

A modell több hozzáférési módot támogat:

- Exclusive Read (ER)
- Concurrent Read (CR)
- Exclusive Write (EW)
- Concurrent Write (CW)

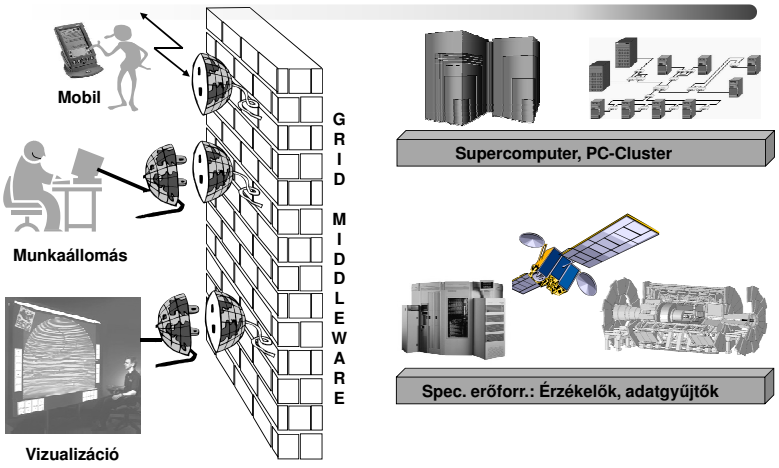
Klasszikus PRAM modellek

- CREW (concurrent read, exclusive write)
  - leginkább használt
- CRCW (concurrent read, concurrent write)
  - legjobb teljesítményű
- EREW (exclusive read, exclusive write)
  - leginkább megszorító
  - legrealisztikusabb
- CROW (concurrent read, owner write)r write
- Common CRCW, Priority CRCW, …

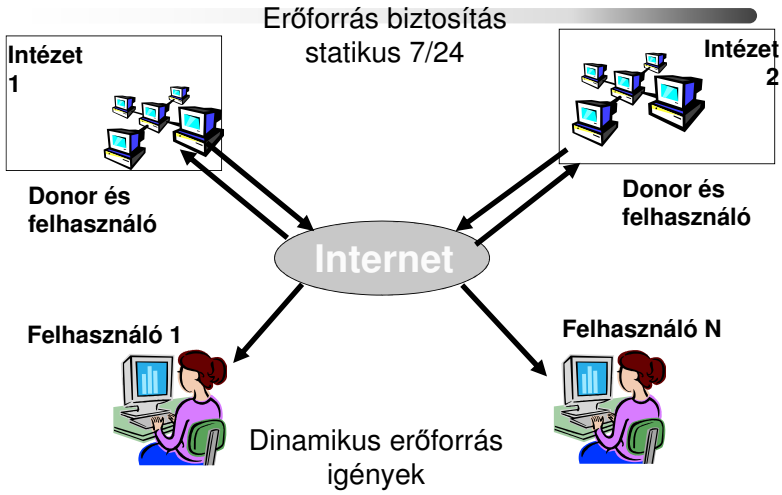
Grid koncepció

- Számítógépek erőforrásainak egy adott cél érdekében összefogott halmaza, melyet a felhasználó egységesen, egy egészként kezelve tud elérni a Grid bármely pontjáról.
- A Grid szóhasználat szándékosan utal az elektromos hálózatra (power grid).
- A kezdeti intézményi gridek regionális, nemzeti, ill. világméretű gridekké nőnek, melyek erőforrásait dinamikusan és gazdaságosan lehet elosztani.
- Adat, számítási és információs gridek.

Grid hasonlat



Utility Grid modell



A Utility Gridek jellemzői

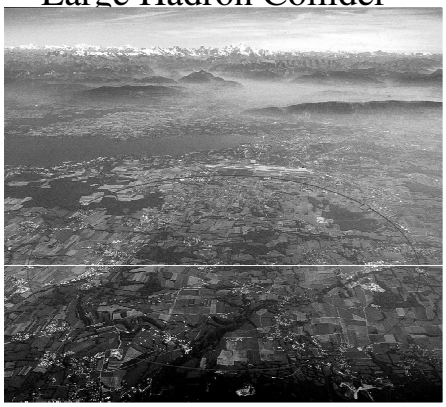
- A donorok profi erőforrás biztosítók (7/24 órás üzemmód) → *Egyszerűsítés*
- Hasonló erőforrások→ *Egyszerűsítés*
- Mindenki használhatja az erőforrásokat saját problémáinak megoldására
- Aszimmetrikus kapcsolat a donorok és felhasználók között

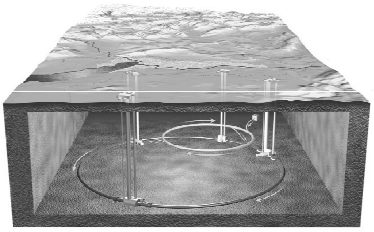
U >> D

LHC

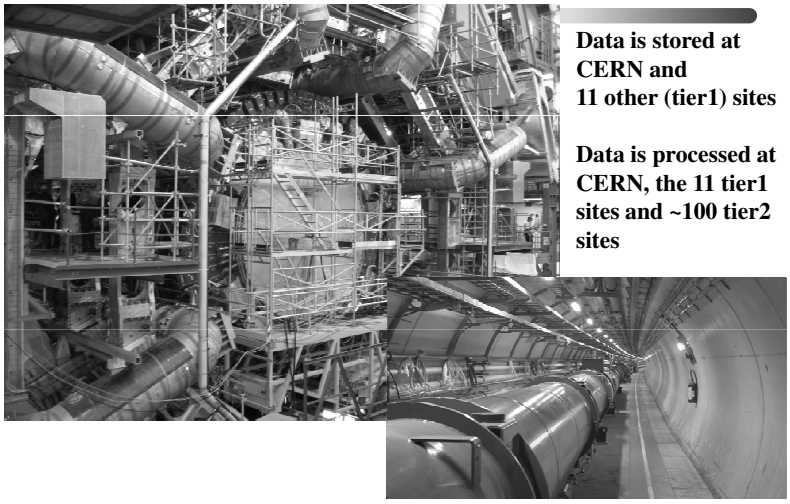
Large Hadron Collider

Produces ~15 PByte/year





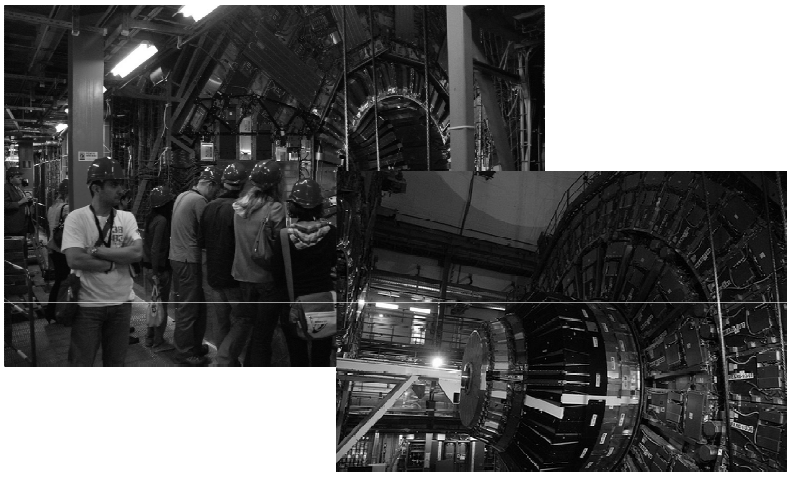
LHC



Data is stored at CERN and 11 other (tier1) sites

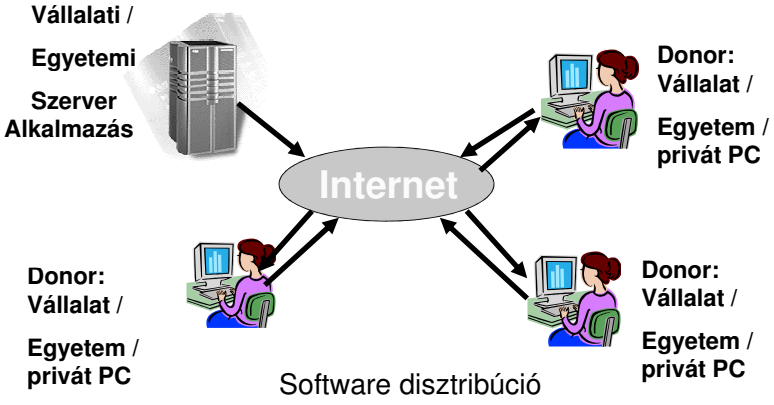
Data is processed at CERN, the 11 tier1 sites and ~100 tier2 sites

Egyik kísérlet (CMS) detektora



Desktop Grid modell

Dinamikus erőforrás biztosítás



Újabb buzzword?

- Metacomputing
- Utility computing
- Grid computing
- IaaS – Infrastructure as a Service
- PaaS – Platform as a Service
- SaaS – Softare as a Service
- ???

Cloud computing def.

- Még bizonytalan a def., többen mást gondolnak róla. NIST definíció:
- A hálózati felhőből on-line igénybe venni
  - számítási, tárolási kapacitást
  - alkalmazást
  - egyéb erőforrást
- Lényegében Web 2.0 kiterjesztve?

Jellemző tulajdonságok (NIST)

1. Igény szerinti önkiszolgálás
   - konfig. lehetőségek
2. Széles hálózati elérés
   - vékony/vastag kliens
3. Erőforrások egyesítése és megosztása
   - több felhasználót (bérlőt) is kiszolgál
4. Rugalmas, gyors konfigurálhatóság
5. Szolgáltatások mérése/számlázása

Cloud rendszerezés

- Szolgáltatási rétegek szerint
  - IaaS
  - PaaS
  - SaaS
  - ??
- Telepítési modell sezrint
  - Privát
  - Publikus
  - Hibrid
  - Közösségi
  - Kormányzati

IaaS

Infrastructure as a Service (computer infrastr.)

- Amazon Web Services
- Rackspace
- Cloud.com
- Openstack
- Terremark
- vCloud

PaaS

Platform as a Service (solution stack)

- App Engine (Google)
- Azure (MS)
- Engine Yard
- Force.com
- Heroku
- S3 (Amazon)
- SQS (Amazon)

SaaS

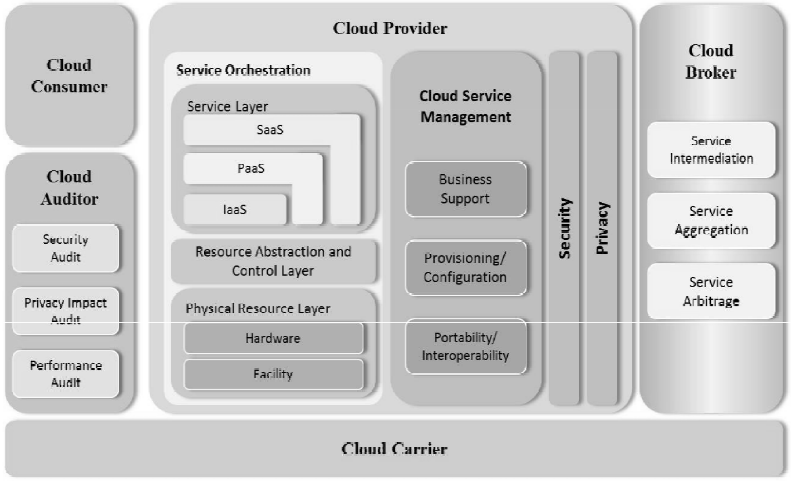
Softare as a Service

- Szoftver alkalmazás igénybevétele web felületen on-line módon
  - Clarizen
    - teljes projektmenedzsment
  - Google Docs
  - SlideRocket
  - Blists
    - database app

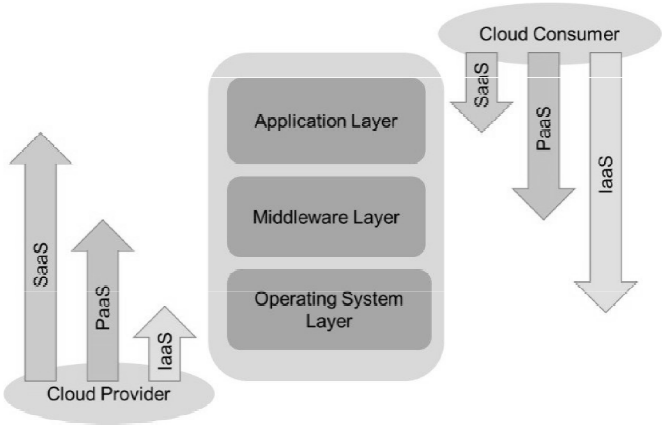
SaaS/2

- Microsoft Hosting, Microsoft Resource Directory
- Oracle on Demand
- IBM Cloud Computing Speciality
- HP Cloud Assure on SaaS
- SAP Cloud

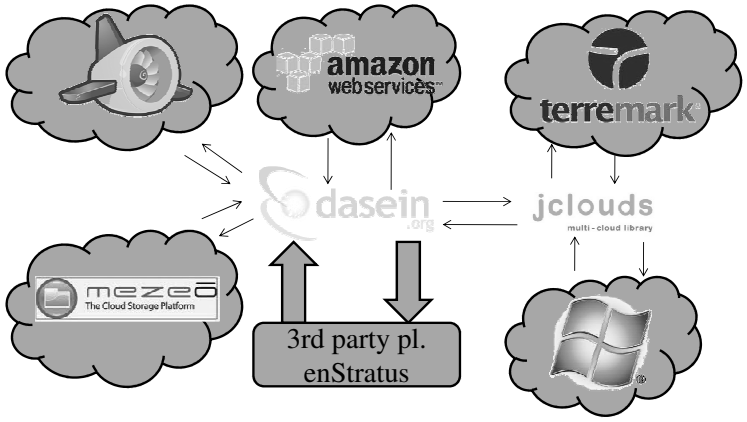
Koncepcionális modell (NIST)



Rétegek közötti kapcsolat (NIST)



A Dasein Cloud API



A Dasein Cloud API

- Java nyelvű, open source (Apache v2.0), aktívan fejlesztett programkönyvtár.
- Számos IaaS szolgáltatót (AWS, Terremark, Rejila), privát felhőt (vCloud, vSphere, CloudStack), storage rendszert (Rackspace, Mezeo, a Google App Engine vagy az MS Azure BlobStore szolgáltatása) kezel.
- Implementációja épít a platform-specifikus megoldásokra (vSphere VIM), és a jclouds open source API-ra.

A Dasein API szolgáltatásai

- Hozzáférés-vezérlés
- Számlázás
- Statikus IP hozzárendelés
- Storage és Content Distribution, Network kezelés
- Adatközpontok kezelése geográfiai elhelyezkedésük szerint
- VM, machine image és virtuális meghajtó kezelés
- Tűzfalak kezelése
- Load balancer és auto-scaling
- Push notification eseménykezelés