# Cache for Workflows

Andrew Harrison

Matthew Shields → *Computer School, Cardiff University, UK*

Ian Taylor

Carlo Mastroianni → *ICAR-CNR, Italy*


Work carried out under the *FP6 Network of Excellence CoreGRID*
funded by the European Commission

# Presentation Outline

- New trends for Internet scale computing
  - Execution of scientific **workflows** on private computers (**public resource computing**)
  - **Super-peer** overlays for routing resource discovery
  - Decentralized **data-caching** schemes
- Alchemist: data caching in super-peer overlays
  - Objective: integrate routing and caching of both adverts and actual data
- Reference applications
  - Distributed cycle sharing applications
  - Workflows for audio-video (e.g., music) analysis
- A super-peer protocol for job execution and data download
  - Performance analysis by simulation

# Public Resource Computing

- The term is used for applications in which jobs are executed by private-owned computers that use their spare CPU time to support a large scientific computing project
- The pioneer project is the well known **SETI@home** (Search for Extra Terrestrial Intelligence) which has attracted millions of participants
- A number of other projects are today supported by the **BOINC** software system (*Berkeley Open Infrastructure for Network Computing*)
  - the **Einstein@home** project aims at detecting certain types of gravitational waves
  - the **Climate@home** focuses on long-term climate prediction

# Super-peers and data caching

- **P2P** algorithms are used today not only for file sharing, but also for distributed computing (SETI@home, BOINC applications)
- The **super-peer** model has been proposed to increase scalability and better match the different capabilities of Internet hosts
  - Super-peer layers define policies for interconnectivity, routing/forwarding, and caching of **adverts**
  - We believe that also **data caching** functionalities could be assigned to super-peers, which are aware of their neighbors and can adopt efficient forwarding policies
  - This would help the **integration** of different functionalities (routing, caching, discovery)

# Alchemist

- P2P framework
  - For decentralized caching of metadata (adverts) **and** application data
- Alchemist uses the super-peer paradigm but
  - Allows different overlays to be created for caching, replication, forwarding
  - Allows workflows or custom code to be deployed and inserted at any part of the system
  - Allows sophisticated Grid-style security sign-on, delegation
- Based on existing technologies
  - P2PS, WSPeer, Triana

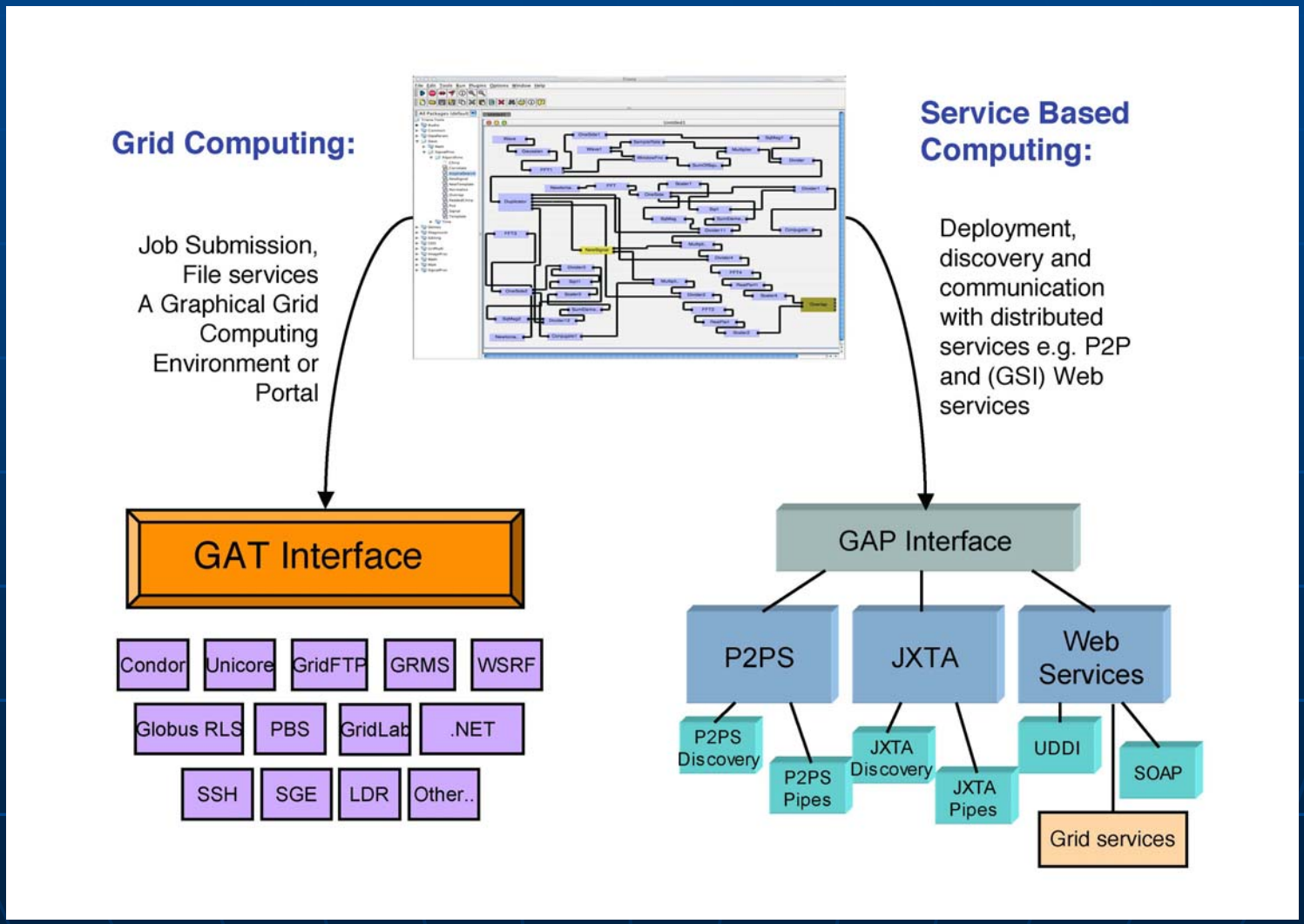
# Core Technologies in Alchemist

- **P2PS**
  - It is a P2P middleware built in Cardiff as an alternative to JXTA
  - P2PS contains much less code than JXTA, but it efficiently supports self-organizing networks and includes multiple endpoint resolvers (TCP, UDP, Multicast, SSL), XML massaging, security functionalities
- **WSPeer**
  - WSPeer allows the creation, publishing, discovery and composition of Web and Grid services
  - Thanks to bindings to JXTA and P2PS, services can be managed as peers and hosted in a P2P environment (allowing to use P2P publishing and discovery functionalities)

# Core Technologies in Alchemist

- **TRIANA**
  - It was originally designed for the **GEO 600** gravitational wave project, to perform on-the-fly analysis of data
  - Then it evolved into a **problem-solving** environment and **workflow management** system
  - Triana is applied in many domains including radio astronomy, data mining, grid-enabled medical applications, biodiversity problems

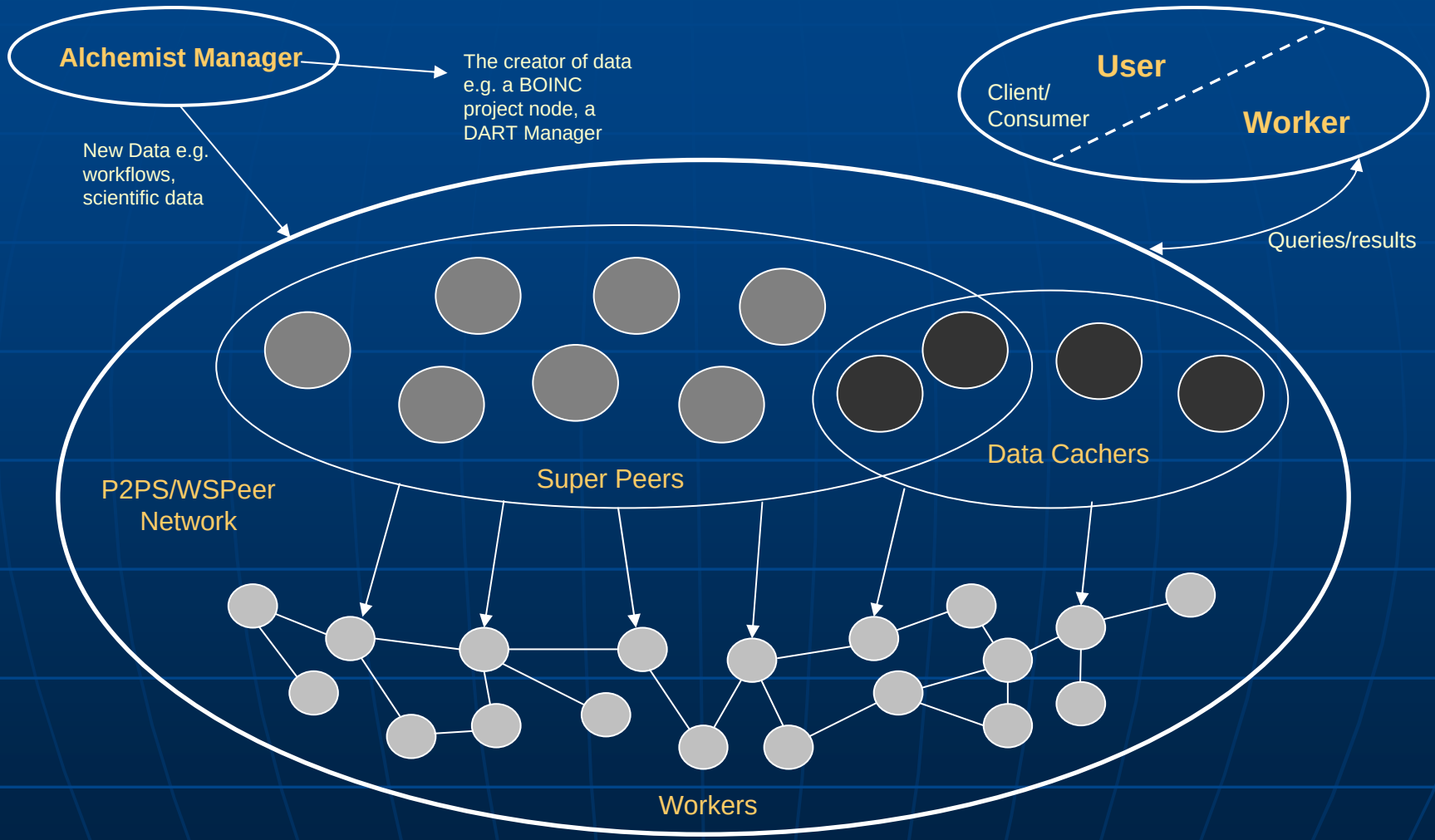
# Triana


Grid Computing:
Job Submission, File services A Graphical Grid Computing Environment or Portal
Service Based Computing:
Deployment, discovery and communication with distributed services e.g. P2P and (GSI) Web services
GAT Interface
Condor
Unicore
GridFTP
GRMS
WSRF
Globus RLS
PBS
GridLab
.NET
SSH
SGE
LDR
Other..
GAP Interface
P2PS
JXTA
Web Services
P2PS Discovery
P2PS Pipes
JXTA Discovery
JXTA Pipes
UDDI
SOAP
Grid services

# Features of Alchemist

- **Service Based P2P middleware**
  - use of P2PS and WSPeer to build networks of Web/Grid services
- **Decentralized caching overlays**
  - design of layers of super peers to provide routing and discovery functionalities and domain specific caching
- **Workflow-based**
  - Alchemist allows **workflows** to be inserted into the network by non-programmers and executed by available workers

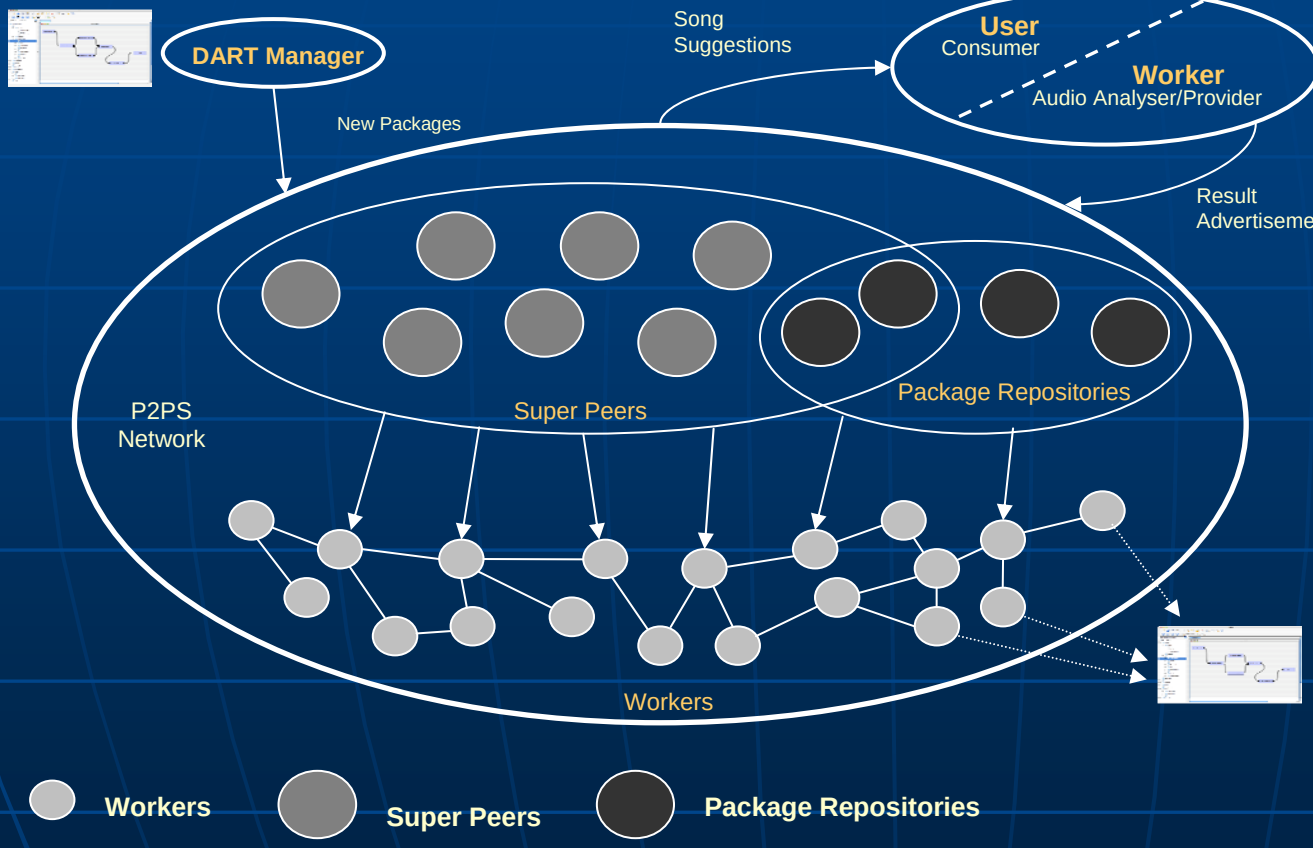
# Overview of Alchemist



# Applications for Alchemist (1)

- DART (Digital Audio Retrieval using Triana)
  - A manager creates a **workflow** to analyze audio (e.g., MP3 files)
  - The workflow is executed by a large number of workers
  - **Metadata**, e.g. results of audio analysis, is returned to the network
  - Applications:
    - **music recommendation** system - i.e. if you like “Band A”, then try “Band B”
    - Complex content based **queries** (e.g., search for tempo, pitch, other features of sound)
    - Example of search operations for tennis matches: find tennis match recording based on referee calls, people cheers etc.

# DART execution with Alchemist




Roles of hosts in DART:

- DART Manager
- Workers
- Super-peers
- Package Repositories (Data Centers)
- ...Users!

# Applications for Alchemist (2)

- **Cycle Sharing (analysis of gravitational waves)**
  - One sample application scenario has been defined for the GridOneD project
  - GridOneD objective: massively distributed analysis of gravitational waveforms produced by binary stars orbiting one around the other
  - Data is analyzed in parallel by a large number of Grid nodes to speed up computation and keep the pace with data production
  - The super-peer paradigm allows to scale the system as the load or the number of users increases
  - Data caching increases performance, as simulation results will show

# A protocol for job execution
## (exploiting super-peer overlays and decentralized caching)

Hosts can assume the different **roles**:

- a job manager node:

  (i) produces the job description files (job adverts), and

  (ii) collects output results

- when workers are available for job execution, they issue job queries to get job adverts and then data queries to collect the corresponding input data files

- super peers play the role of rendezvous nodes, since they can store job and data adverts and compare them with job and data queries

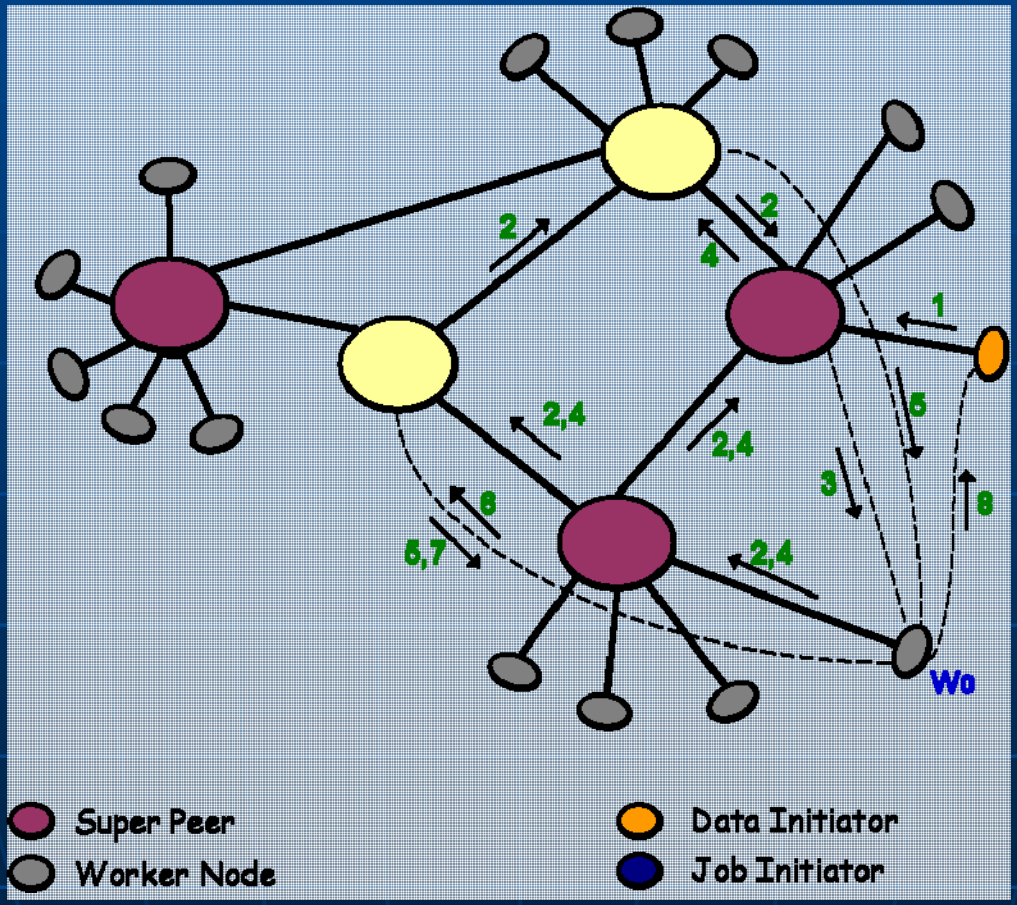
- data centers are nodes capable of caching data files and delivering them to workers

# Matching of jobs and data

The protocol includes **two matching phases**:

- **Job assignment phase**
  - a job query issued by a worker travels across super-peers and is compared with job adverts
  - matching succeeds when job query parameters (*e.g. CPU time and memory amount that the node offers*) are compatible with job advert information (*e.g. characteristics of the platforms on which the job must be executed, information about the required input data file*)
- **Data downloading phase**
  - a worker issues a data query to discover a matching data advert, then downloads actual data file from a data center
  - this phase is made more efficient with the presence of multiple data centers

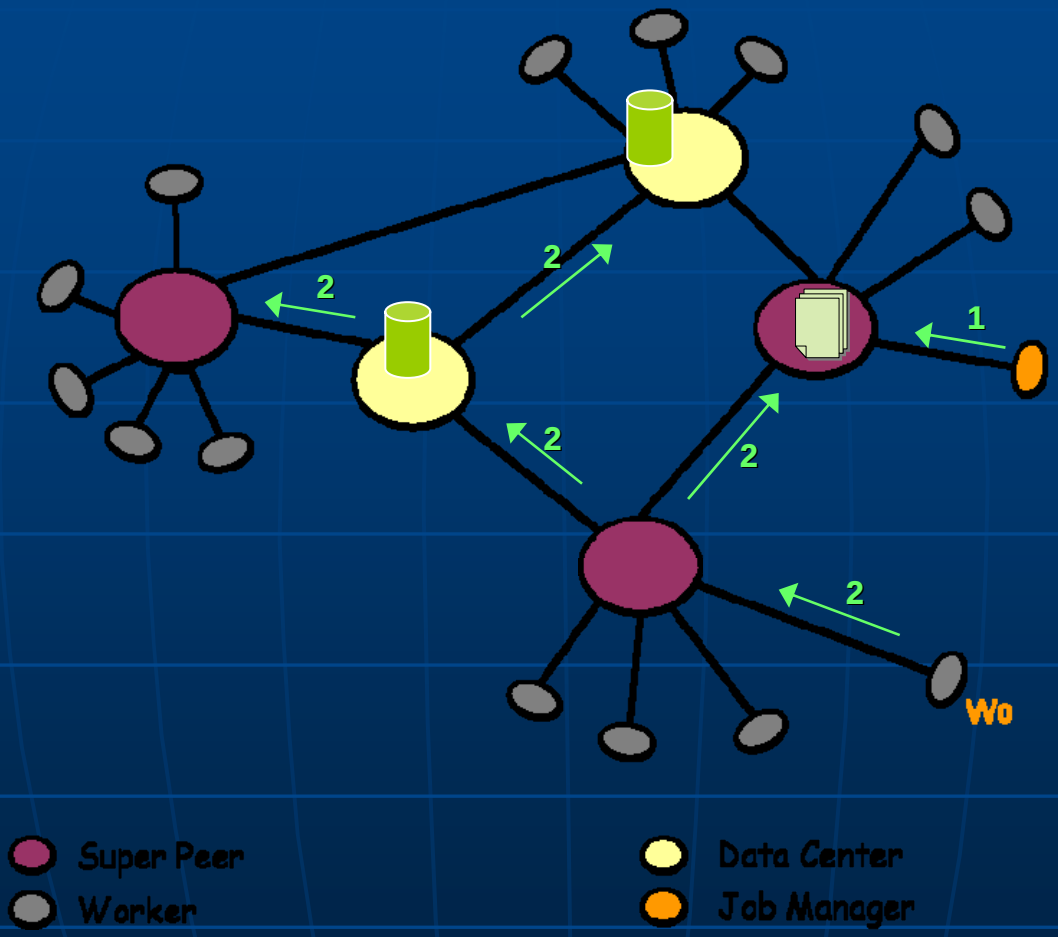
# Protocol behavior in a sample network with 5 super-peer and 2 data centers




*1. job advert*
*2. job query*
*3. job assignment*
*4. data query*

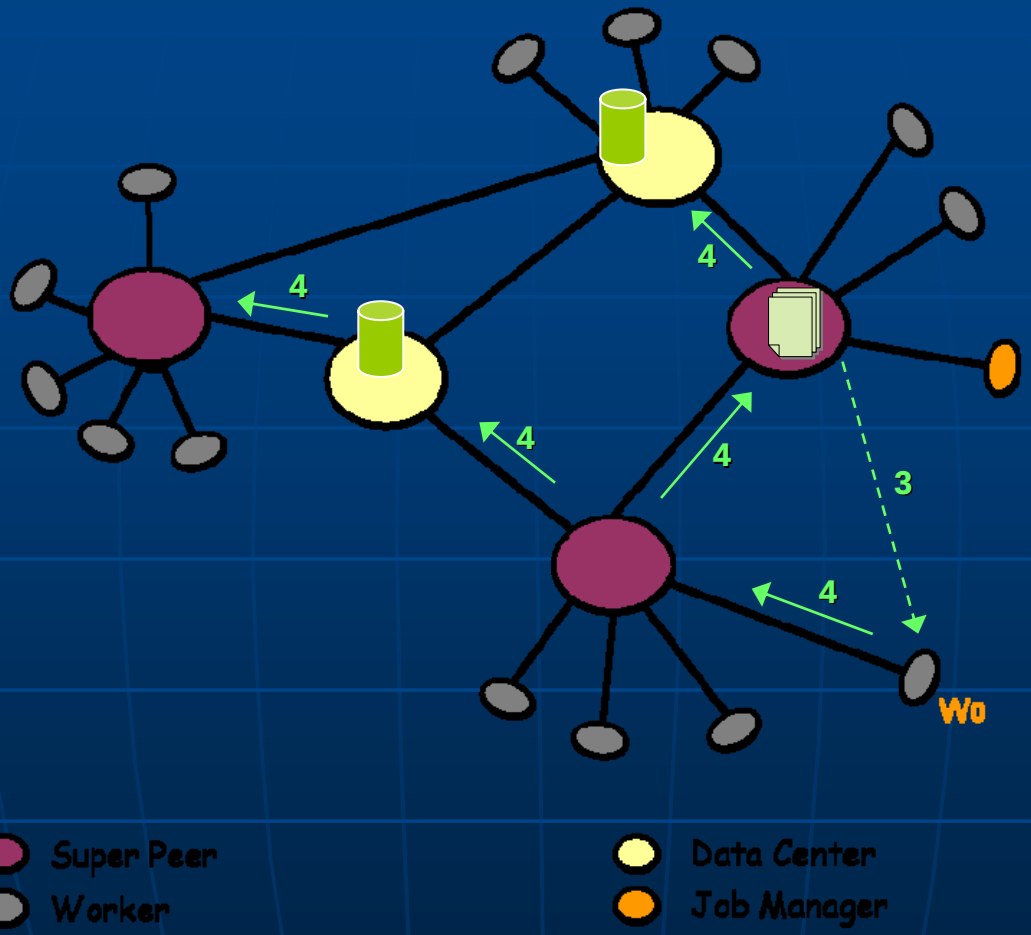
*5. data advert*
*6. data download request*
*7. data download*
*8. job results*

# Protocol behavior in a sample network with 5 super-peers and 2 data centers



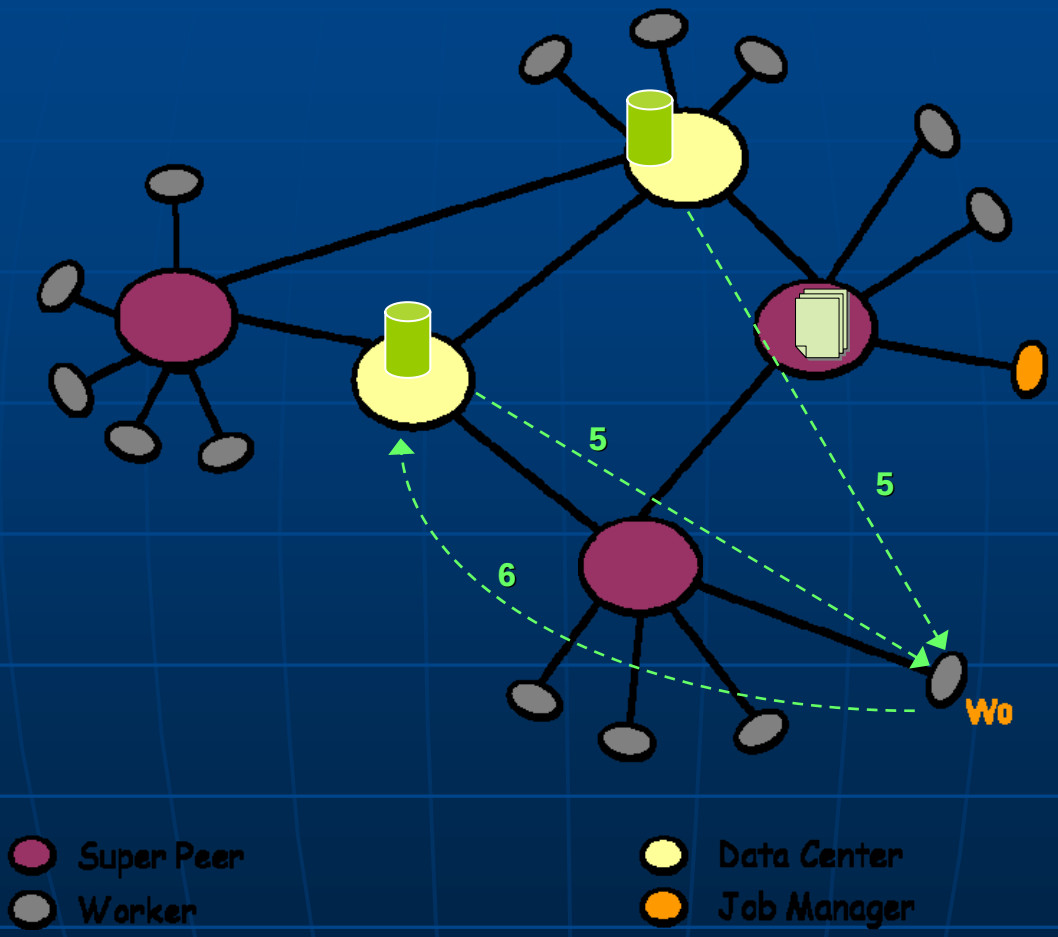

1. **Job advert.** The job manager generates the job adverts describing the features of the jobs that must be executed. The job adverts are stored by the neighbor super-peer
2. **Job query.** The worker $W_0$ issues a job query that is forwarded through the super-peer network until it reaches a matching job advert

# Protocol behavior for a sample network with 5 super-peers and 2 data centers



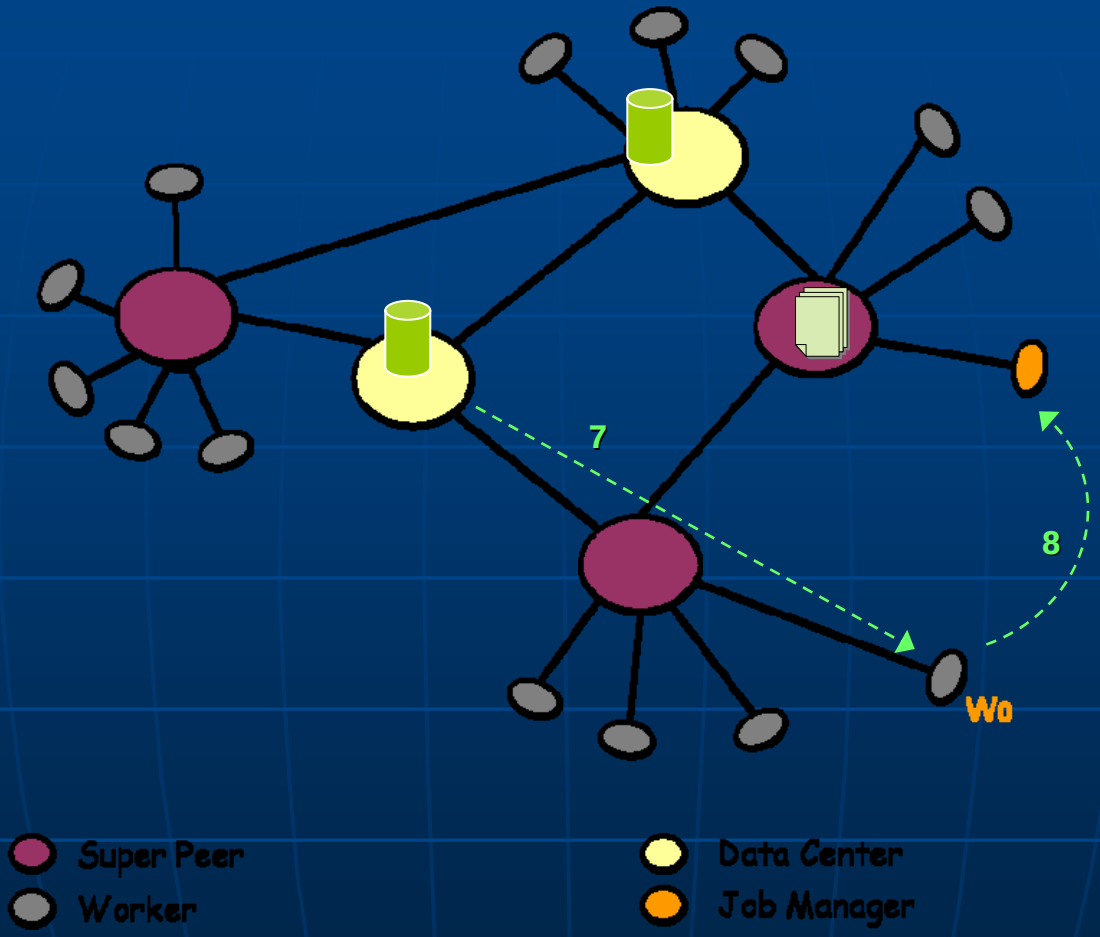

3. **Job assignment.** When the job query finds a matching job advert on a super-peer, the latter sends a job assignment directly to the worker that issued the query
4. **Data query.** The worker inspects the job advert and issues a data query for the input data file needed to execute the job. The data query is forwarded through the super-peer network until it finds one or more data centers that have the data file

# Protocol behavior for a sample network with 5 super-peers and 2 data centers




5. **Data advert.** The data centers which match the data query send a data advert message to the worker
6. **Data download request.** The worker selects the most convenient data center, and sends it a download request

# Protocol behavior for a sample network with 5 super-peers and 2 data centers




7. **Data download.** The worker downloads the input data file (the file is also cached by the local super-peer if it is a data center)
8. **Job execution and results.** The worker executes the job and after its completion sends the results to the job manager. Now the worker can issue a new job query

# Sample application: analysis of astronomical data

- GridOneD project (Cardiff): massively distributed search and analysis of gravitational waveforms produced by binary stars orbiting one around the other
- A file of about 7.2 MB of data is produced every 15 minutes and it must be compared with a large number of templates (between 5,000 and 10,000) by performing fast correlation
- Data can be analyzed in parallel by a number of Grid nodes to speed up computation and keep the pace with data production

**Other possible application: Distributed Audio Retrieval using Triana**

- In this case data consists of Triana workflow packages containing tools and process logic

# Scenario and Parameters

- We analyzed this sample scenario
  - 25 super-peers + 250 worker nodes
  - Maximum number of neighbor of a super-peer = 4
  - 7.2 MB data file split in 100 KB fragments for downloading
  - Local and remote latencies: respectively 10 ms and 100 ms
  - Local and remote bandwidths: respectively 10 Mbps and 1 Mbps
  - Mean job processing time: 500 s
  - *Simplifying assumption: data centers have received data before the job submission phase (current research focuses on on-the-fly caching)*

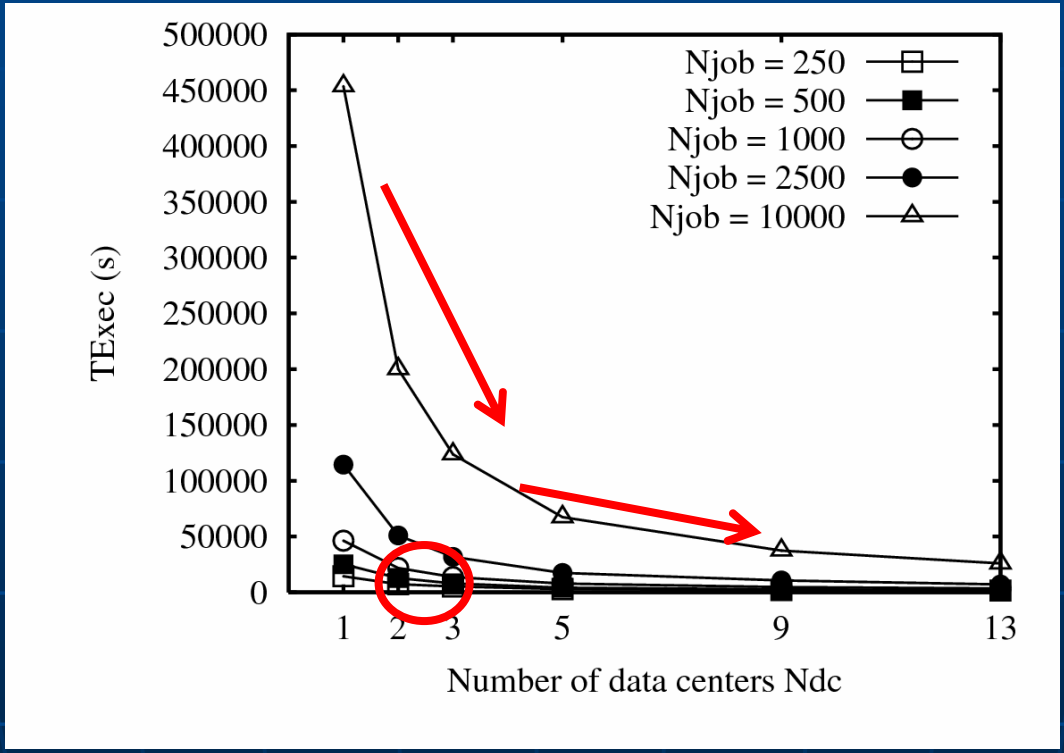
- The following parameters were given variable values
  - Number of jobs **Njob**: from 250 to 10000
  - Number of data centers **Ndc** = 1,2,3,5,9,13

# Performance Indices

- Overall execution time Texec
  - time (sec) needed to execute all the jobs
- Percentage of activity time Pact
  - average percentage of time in which a Data Center is active, i.e. it has at least one download connection in progress
- Max number of executed jobs Jmax
  - maximum number of jobs executed by a single worker
  - it is useful to evaluate load balancing among workers

# Overall execution time

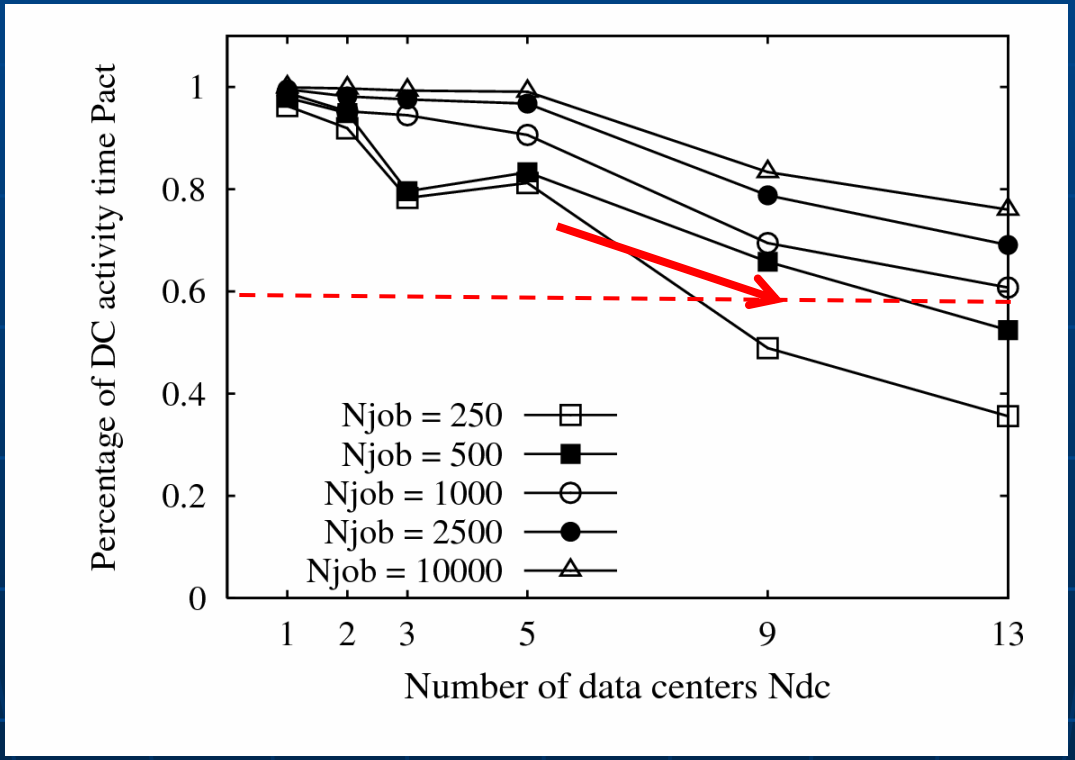
Overall execution time Texec vs. the number of data centers Ndc, for different values of the number of jobs Njob




- The overall execution time decreases as more data centers are made available
- It is possible to determine an appropriate number of data centers, depending on the number of jobs:
  - *with 10,000 jobs, a significant reduction of Texec is perceived as the number of data centers is increased up to 9*
  - *with 1,000 jobs or less, two or three data centers are sufficient*

# Activity of Data Centers

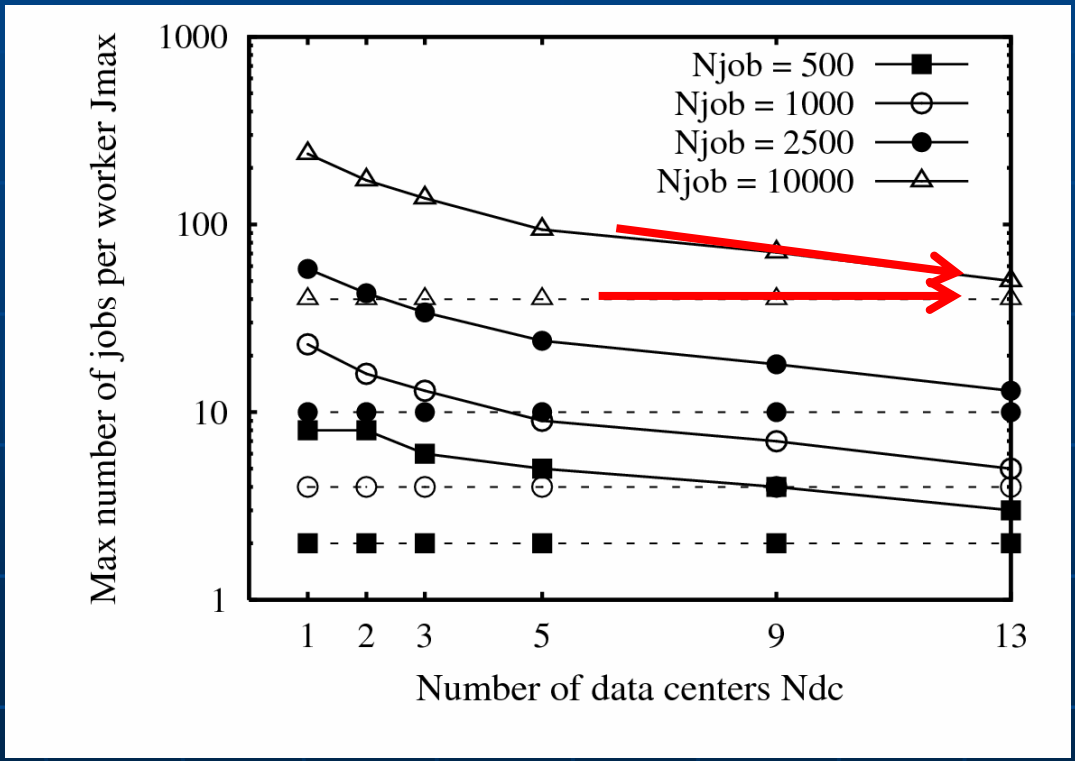
Percentage of activity time of a data center Pact vs. the number of data centers Ndc, for different values of the number of jobs Njob




- Results show that the presence of an excessive number of data centers can be inappropriate, especially if the number of jobs is not very large
- Indeed when the percentage of activity time decreases below 60%, machine utilization is low, possibly resulting in a poor return of investment (ROI)

# Load balancing among workers

Maximum (Jmax) and average number of jobs vs. the number of data centers Ndc, for different values of the number of jobs Njob




- This figure compares the number of jobs executed by a worker on average (obtained as Njob/250) to the maximum number of jobs executed by a single worker
- It is interesting to note that the two indices approach one another as the number of data centers is increased, leading to a fairer load balancing among workers

# Final remarks

- We discussed current super peer overlays and how they can be used to form data overlays for the efficient and scalable distribution of data
- We presented the Alchemist framework, that :
  - efficiently integrates different functionalities: caching of adverts and data, matching, routing
  - supports definition and execution of workflows
- We proposed a super-peer protocol for multiple job submission, which exploits the presence of data centers
- We analyzed the performance of this protocol by event-driven simulation
  - Analysis suggests that the protocol can be efficient and scalable, if the number of data centers is properly tuned

# Current Research

- Current research is moving along a number of interesting research avenues, such as:
  - analysis of redundant computing for applications that require several executions of each job
  - progressive caching of data file fragments on the super-peer network to improve data download performance (BitTorrent style)
  - performance analysis of the super-peer protocol in the case that data is progressively fed by an external source as a data stream

# Thanks !

Links:

- http://www.trianacode.org/
- http://www.wspeer.org/
- http://www.trianacode.org/p2ps/
- http://www.mrsdart.com/