

Issues on High-Performance Searching for Heterogeneous Data Collections on the GRID

Diego Puppin

PhD Candidate
Università di Pisa

June 7, 2006



Outline

- 1 The Story So Far**
 - Components and Web Services
 - Distributed Information Systems/P2P
 - A Prototype Search Engine for Components
- 2 What's Next
- 3 The Query-vector Model
- 4 Experiments
- 5 Conclusions
 - Results and Publications



Goal

High-performance searching tool for components, services and resources on the GRID, from independent organizations/providers

GRID Definition by CoreGrid Ex. Comm.

A fully distributed, dynamically reconfigurable, scalable and autonomous infrastructure to provide location independent, pervasive, reliable, secure and efficient access to a coordinated set of services encapsulating and virtualizing resources (computing power, storage, instruments, data, etc.) in order to generate knowledge.



Analysis of Component Systems

- Initial analysis of existing systems to get some background
 - CCA, XCAT, CCM, Fractal
- Compared qualitative and quantitative aspects
- Developed some tools to manipulate Fractal/CoreGrid GCM in the prototype search engine (later)

Related Publications

- Aldinucci M., Campa S., Coppola M., Danelutto M., Laforenza D., Puppini D., Scarponi L., Vanneschi M. Components for high-performance grid programming in Grid.IT "Component Models and Systems for Grid Applications", edited by Vladimir Getov and Thilo Kielmann, Springer, 2004
- Puppini D., Silvestri F., Laforenza D. An evaluation of component-based software design approaches. CCGRID 2004. 4th IEEE/ACM International Symposium on Cluster Computing and the Grid (Chicago, USA, 19-22 April, 2004).

Performance Issues of Grid/Web Services

- Studied the feasibility of GSs/WSs to implement HP computing
- Positive results when porting an MPI application to WS
- Very simple and effective component model
- GSs/WSs are a valid/searchable component model

Related Publications

- Puppini D., Tonello N., Laforenza D. Using web services to run distributed numerical applications. 11th EuroPVM/MPI 2004 (Budapest, 19-22 settembre 2004). Springer LNCS 3241/2004
- Puppini D., Silvestri F., Laforenza D. Component Metadata Management and Publication for the Grid in Proceedings of the International Conference on Information Technology Coding and Computing ITCC 2005, Las Vegas, NV, USA, 4-6 April, 2005

P2P Information System

- P2P information system to address reliability and scalability
- Strong potential for a Grid Information System
- Initial solution based on routing indexes
- Compared with centralized IS and GT3/GT4 IS
 - Much faster, fresher data
- Still missing a strong selection/distribution strategy

Related Publications

- Puppini D., Moncelli S., Baraglia R., Tonello N., Silvestri F. A Grid Information Service Based on Peer-to-Peer in Proceedings of EuroPar2005. Springer LNCS 2648/2005.



A Search Engine for Components

- Key tool for Grid programming
- Starting points for a IDE with composition, profiling etc.
 - Discussed its possible inclusion in GridComp
- Created an initial prototype



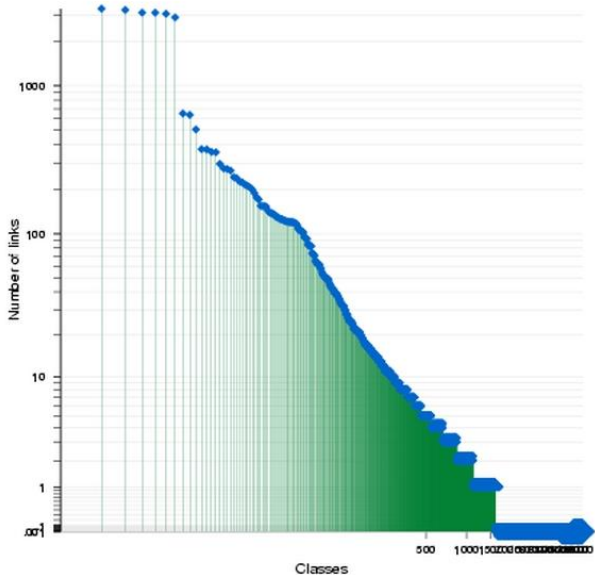
Using the Web Search Technology

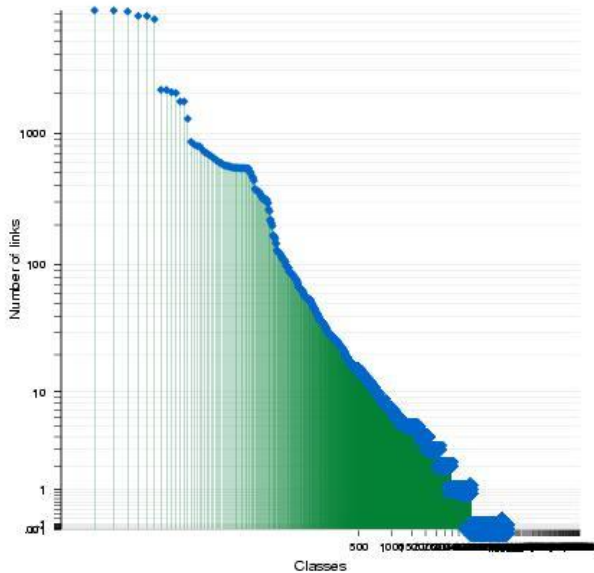
- Observed a power-law distribution of inlinks.
- ...like the Web.
- This is a natural emerging property, from independent projects.
- Independent developers shape a scale-free ecosystem.
- Java class usage behaves like Web pages linking
 - Same power-law distribution
- This justified the use of Page Rank for classes

Related Publications

- Puppini D., Silvestri F. The Social Network of Java Classes SAC '06, April 23-27, 2006, Dijon, France.







Class Rank

To determine the rank of a class C, we iterate the following formula:

$$\text{rank}_C = \lambda + (1 - \lambda) \sum_{i \in \text{inlinks}_C} \frac{\text{rank}_i}{\#\text{outlinks}_i}$$

where inlinks_C is the set of classes that use C (with a link into C), $\#\text{outlinks}_i$ is the number of classes used by i (number of links out of i), and λ a small factor, usually around 0.15.



Top-ranking classes

- String, Object, Class, Exception
- #7: Apache MessageResources
- #11: Tomcat CharChunk
- #14: DBXML Value
- #73: JXTA ID



Ranking better than Pure TF.IDF

TF.IDF for “file writer”:

- 1 javax.jnlp.JNLPRandomAccessFile, (JNLP API Reference 1.5);
- 2 javax.swing.filechooser.FileSystemView (J2SE 5.0);
- 3 java.io.FileOutputStream (J2SE 5.0);
- 4 java.io.RandomAccessFile (J2SE 5.0).

Class Rank for “file writer”:

- 1 java.io.PrintWriter;
- 2 java.io.PrintStream;
- 3 java.io.File;
- 4 java.util.Formatter.

all from Java API (J2SE 5.0).



GRIDLE 0.1

- Ranking using two metrics:
 - TF.IDF (term frequency times inverted document frequency)
 - GRIDLE Rank
- Bells and whistles:
 - Snippets, Links and Reverse Links
- <http://gridle.isti.cnr.it>



A search engine for SW components



GRIDLE: Google-like Ranking, Indexing and Discovery service for a Link-based Eco-system of software components



http://gridle.isti.cnr.it/?query=file+buffer&numero=10&sort=pagerank&submit=submit



Find high-relevance Java classes out of a repository of **7700 elements!!!**

QUERY:

RESULTS:

Sort by [Class Rank](#) or [TF.IDF](#)

SORTED BY CLASS RANK



[PrintWriter \(Java 2 Platform SE 5.0\)](#)

Class Path: java.io.PrintWriter

... The output will be written to the **file** and is **buffered**.csn - The name of a supportedHREF=".../java/nio/charset/Charset.html" title="class in java.nio.charset"> charsetThrows: HREF=".../java/io/ **file**NotFoundException.html" title="class in java.io"> **file**NotFoundException - If the given string does not denote an existing, writable regular **file** and a new regular **file** of that name cannot be created, or if some

...
<http://java.sun.com/j2se/1.4.2/docs/api/java/io/PrintWriter.html>

Score: 35.75 - [Cached copy](#) - [Class Graph](#)



[PrintStream \(Java 2 Platform SE 5.0\)](#)

Class Path: java.io.PrintStream

... The output will be written to the **file** and is **buffered**.csn - The name of a supportedHREF=".../java/nio/charset/Charset.html" title="class in java.nio.charset"> charsetThrows: HREF=".../java/io/ **file**NotFoundException.html" title="class in java.io"> **file**NotFoundException - If the given **file** object does not denote an existing, writable regular **file** and a new regular **file** of that name cannot be created, or if ...

...
<http://java.sun.com/j2se/1.4.2/docs/api/java/io/PrintStream.html>

Score: 35.50 - [Cached copy](#) - [Class Graph](#)

Prototype GRIDLE

- It also indexed Fractal component
- International collaboration with the Proactive/Fractal group at INRIA-Sophia Antipolis

Related Publications

- Silvestri F., Puppini D., Laforenza D., Orlando S. Toward a search engine for software components. IEEE Web Intelligence (Beijing, China, September 20-24, 2004). IEEE, 2004.
- Diego Puppini, Matthieu Morel, Denis Caromel, Domenico Laforenza, Françoise Baude GRIDLE Search for the Fractal Component Model. To appear in Proceedings for the Pisa CoreGRID integration workshop, Pisa, November 2005, published by Springer-Verlag

Wrapping Up The Story So Far

- Showed that some existing component models offer high performance and are easily searchable
 - Quantitative analysis of different model, emphasis on WS
- Argued that a distributed information system can solve the problem of finding components/services/resources on the Grid
 - Better availability, scalability, freshness
- Noticed that component usage follows the same pattern of the Web
 - Power link distribution of links - Class rank
 - Integration with Fractal



Outline

- 1 The Story So Far
 - Components and Web Services
 - Distributed Information Systems/P2P
 - A Prototype Search Engine for Components
- 2 What's Next**
- 3 The Query-vector Model
- 4 Experiments
- 5 Conclusions
 - Results and Publications



When on the Grid

- Need for a distributed search tool
- Resources/Services/Components are provided by independent organizations
- The search engine must collect and index data from heterogeneous providers
- Scalability issues, with very large test-beds
- For every query, we must route the query to the most suitable server
 - Both if we have to query independent providers
 - Or if we choose to partition the data on several servers
- Key problem also for P2P solutions



How to Build a Distributed Information Service

- Resources are indexed according to their description (document)
- We build a resource-term matrix storing the presence of terms in the description
- This can be boolean or weighted (for more rare terms)
- Then we partition the index
 - Lot of results for Web search engines



Example

	res1	res2	res3	res4	res5	res6
term1		0.5	0.8	0.1		0.2
term2	0.3		0.2		0.1	0.8
term3	0.1	0.5	0.8			
term4	0.2	0.5		0.2		0.5
term5		0.1	0.1			
term6				0.3	0.6	0.3



Example: Document Partitioned

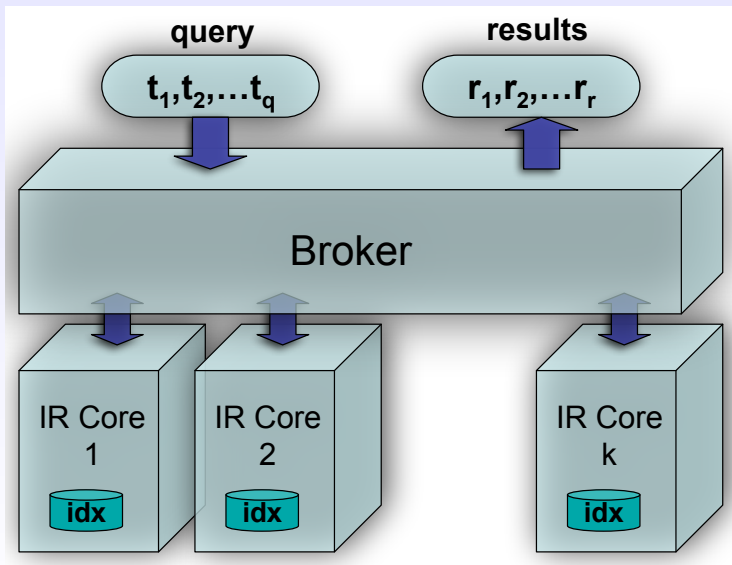
	res1	res2	res3	res4	res5	res6
term1		0.5	0.8	0.1		0.2
term2	0.3		0.2		0.1	0.8
term3	0.1	0.5	0.8			
term4	0.2	0.5		0.2		0.5
term5		0.1	0.1			
term6				0.3	0.6	0.3



Example: Term Partitioned

	res1	res2	res3	res4	res5	res6
term1		0.5	0.8	0.1		0.2
term2	0.3		0.2		0.1	0.8
term3	0.1	0.5	0.8			
term4	0.2	0.5		0.2		0.5
term5		0.1	0.1			
term6				0.3	0.6	0.3





Term-partitioned Index

- Terms are assigned to servers
- Queries are submitted only to servers holding the relevant terms
- **Only a subset of servers is queried**
- Results from each server are intersected/merged and ranked
- **Can reduce the overall system load**



Document-partitioned Index

- Resource descriptions are assigned to servers
- A query can be submitted **to each cluster, to improve throughput**
- ... OR ... **to reduce load, only to selected servers**
- We must choose the “good servers” in advance
- Problem of partitioning and collection selection



CORI

- State-of-the-art for collection selection
- It stores
 - $df_{i,k}$, the number of documents in collection i containing term k , which is $O(dc \times t)$ (before compression),
 - cw_i , the number of different terms in collection i , $O(dc)$,
 - cf_k , the number of resources containing the term k , $O(t)$.
- These data are used in a simple formula to rank documents



Several Approaches to Partitioning and Selection

Document partitioning:

- Document clustering with k-means
- Semantic clustering with directories
- Random/round robin

Collection Selection:

- CORI
- Random
- All collections are queried
- Online sampling

Now, let's try something new!



k-means

- Iterative method to cluster documents
- We create an initial random allocation
- For each group, we compute the average vector
- We assign each document, in turn, to the cluster with the closest average
- Very simple but very computing-intensive



Outline

- 1 The Story So Far
 - Components and Web Services
 - Distributed Information Systems/P2P
 - A Prototype Search Engine for Components
- 2 What's Next
- 3 The Query-vector Model**
- 4 Experiments
- 5 Conclusions
 - Results and Publications



Two Birds with One Stone

- Trying to make clusters of resources that answer to similar query
- Also trying to clusters queries that recall similar resources
- Let's co-cluster [Dhillon 2003] the query-resource matrix
- Very fast algorithm (much faster than k-means)



Cocustering Example

$$p(X, Y) = \begin{bmatrix} .05 & .05 & .05 & 0 & 0 & 0 \\ .05 & .05 & .05 & 0 & 0 & 0 \\ 0 & 0 & 0 & .05 & .05 & .05 \\ 0 & 0 & 0 & .05 & .05 & .05 \\ .04 & .04 & 0 & .04 & .04 & .04 \\ .04 & .04 & .04 & 0 & .04 & .04 \end{bmatrix}$$

$$p(\hat{X}, \hat{Y}) = \begin{bmatrix} .3 & 0 \\ 0 & .3 \\ .2 & .2 \end{bmatrix}$$

Rows and columns are shuffled to minimize loss of information.



New Approach

- For every training query, we store the first 100 results of a reference search engine (centralized index)
- We create a query-document matrix, entries proportional to rank
- We co-cluster to put 1's and 0's together (actually, float numbers)
- We create N document clusters and M query clusters
- The process minimizes the loss of information between the original and the clustered matrix

$$\bullet \hat{P}(qc_a, dc_b) = \sum_{i \in qc_b} \sum_{j \in dc_a} r_{ij}$$



Query-vector Representation

For each query, we store the Top-100 results with rank

Query/Doc	d1	d2	d3	d4	d5	d6	...	dn
q1	-	0.5	0.8	0.4	-	0.1	...	-
q2	0.3	-	0.2	-	-	-	...	0.1
q3	-	-	-	-	-	-	...	-
q4	-	0.4	-	0.2	-	0.5	...	0.3
...
qm	0.1	0.5	0.8	-	-	-	...	-

We may have **empty** columns (resources never recalled, **d5**) and empty rows (queries with no results, **q3**). They are removed before co-clustering. About 52% of resources are recalled by NO query - we can put them in an *overflow* cluster.



Collection Selection using PCAP

- Create big *query dictionaries* by chaining together all the queries from one query-cluster
- Index the dictionaries as documents
- For a new query q , choose the best query-clusters with TF.IDF
 - For each query-cluster qc_i , we get a rank $r_q(qc_i)$
- Compute the rank of each document-cluster:

$$r_q(dc_j) = \sum_i r_q(qc_i) \times \hat{P}(i, j)$$

- The overflow IR core is always queried as the last one



PCAP Example

	dc1	dc2	dc3	dc4	dc5	Rank for q
qc1		0.5	0.8	0.1		0.2
qc2	0.3		0.2		0.1	0.8
qc3	0.1	0.5	0.8			0

Query q ranks the qc respectively 0.2, 0.8 and 0.

$$r_q(dc_1) = 0 \times 0.2 + 0.3 \times 0.8 + 0.1 \times 0 = 0.24$$

$$r_q(dc_2) = 0.5 \times 0.2 + 0 + 0 = 0.10$$

$$r_q(dc_3) = 0.8 \times 0.2 + 0.2 \times 0.8 + 0 = 0.32$$

$$r_q(dc_4) = 0.1 \times 0.2 + 0 + 0 = 0.02$$

$$r_q(dc_5) = 0 + 0.1 \times 0.8 + 0 = 0.08$$

Clusters will be chosen in the order dc3, dc1, dc2, dc5, dc4.



Outline

- 1 The Story So Far
 - Components and Web Services
 - Distributed Information Systems/P2P
 - A Prototype Search Engine for Components
- 2 What's Next
- 3 The Query-vector Model
- 4 Experiments**
- 5 Conclusions
 - Results and Publications



How to Test This Approach

- I could not access a big repository of resources, components or services
 - I tried Fractal, Provenance, NextGrid
- There is no reference benchmark for result quality
- Performed test on the Web
 - Query-log to a Web search engine, documents from a Web snapshot



Data Statistics

<i>dc</i> :	no. of document clusters	16 + 1
<i>qc</i> :	no. of query clusters	128
<i>d</i> :	no. of documents	5,939,061
	total size	22 GB
<i>t</i> :	no. of unique terms	2,700,000
<i>t'</i> :	no. of unique terms in the query dictionary	74,767
<i>tq</i> :	no. of unique queries in the training set	190,057
<i>q1</i> :	no. of queries in the first test set	194,200
<i>q2</i> :	no. of queries in the second test set	189,848
<i>ed</i> :	empty (not recalled) documents	3,128,366

Table: Statistics about collection representation. Data and query-logs from WBR99.



Benchmarks

Partitions based on document contents:

- Random allocation
- Clusters with shingles
 - Signature of 64 permutations
- URL sorting

Partitions based on query-vector representation:

- Clustering with k-means
- Co-clustering (*)

(*) We can use PCAP in this case!

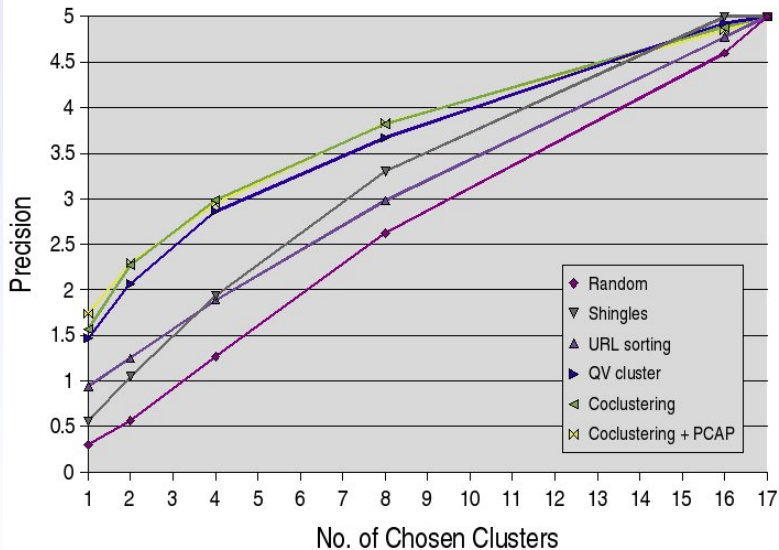


Shingles

- Choose 64 locality-preserving fingerprint functions
 - Similar documents will have similar finger-print
- A document is represented by 64 unsigned long integers
- Perform clustering on them



Precision at 5



Precision with one cluster

random allocation (CORI)	0.3
clustering with shingles (CORI)	0.56
URL sorting (CORI)	0.94
clustering with k-means on query-vectors (CORI)	1.47
co-clustering (CORI)	1.57
co-clustering (PCAP)	1.74

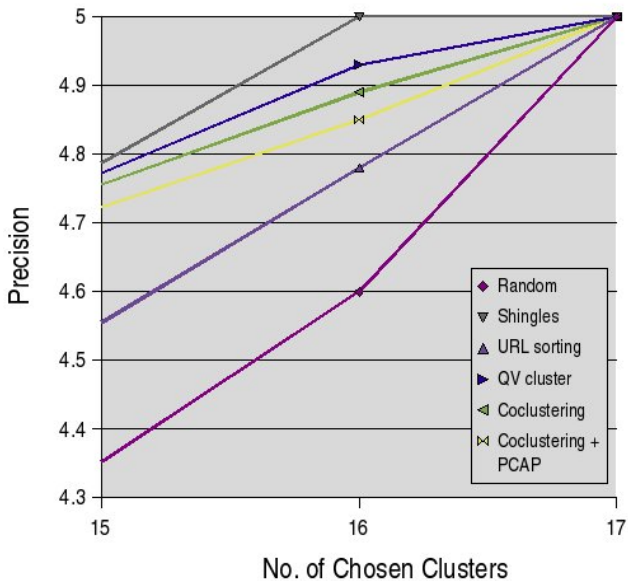
Table: Precision at 5 on the first cluster.



Impact

- If a given precision is expected, we can use FEWER servers
- With a given number of servers, we get HIGHER precision
 - Confirmed with different metrics
- Smaller load for the IR system, with better results
- *No load balancing (for now)*
- 50% of pages contribute to 97% precision
 - We can remove the rest





Robustness to Topic Drift

Results do not change significantly if test with later queries.

Precision at	FOURTH WEEK					
	1	2	4	8	16	17
5	1.74	2.30	2.95	3.83	4.85	5.00
10	3.45	4.57	5.84	7.60	9.67	10.00
20	6.93	9.17	11.68	15.15	19.31	20.00

Precision at	FIFTH WEEK					
	1	2	4	8	16	17
5	1.73	2.26	2.89	3.76	4.84	5.00
10	3.47	4.51	5.75	7.50	9.66	10.00
20	6.92	9.02	11.47	14.98	19.29	20.00

Table: Precision at 5 of the PCAP strategy, on the 4th and the 5th week.



Representation Footprint

CORI representation includes:

- $df_{i,k}$, the number of documents in collection i containing term k , which is $O(dc \times t)$ (before compression),
- cw_i , the number of different terms in collection i , $O(dc)$,
- cf_k , the number of resources containing the term k , $O(t)$.

Total: $O(dc \times t) + O(dc) + O(t)$ (before compression)

dc , number of document clusters (16+1)

t , number of distinct terms, 2,700,000



Representation Footprint (2)

The PCAP representation is composed of:

- the PCAP matrix, with the computed \hat{p} , which is $O(dc \times qc)$,
- the index for the query clusters, which can be seen as $n_{i,k}$, the number of occurrences of term k in the query cluster i , for each term occurring in the queries — $O(qc \times t')$.

TOTAL: $O(dc \times qc) + O(t' \times qc) = 9.4M$ (uncompressed)

CORI: $O(dc \times t) + O(dc) + O(t) = 48.6M$ (uncompressed)

dc , number of document clusters, 16+1

qc , number of query clusters, 128

t' , number of distinct terms in the query dictionary, 74,767

t , number of distinct terms, 2,700,000



Results

- New (smaller) document representation as query-vectors
 - 2.7 M terms vs. 190 K queries
 - More effective on clustering (k-means)
 - Helps with the curse of dimensionality
- New partitioning strategy based on co-clustering
 - Very quick running time
- New (smaller) collection representation based on PCAP matrix
 - About 19% in size before compression
- New strategy PCAP for collection selection
 - 10% better than CORI on different metrics
- Removal of 50% of rarely-asked-for documents with minimal loss
 - They contribute only to 3% of recalled documents



Related Publications

- Puppini D., Silvestri F., Laforenza D. Query-Driven Document Partitioning and Collection Selection Infoscience 2006, May 30-June 1, 2006, Hong Kong.



Outline

- 1 The Story So Far
 - Components and Web Services
 - Distributed Information Systems/P2P
 - A Prototype Search Engine for Components
- 2 What's Next
- 3 The Query-vector Model
- 4 Experiments
- 5 **Conclusions**
 - Results and Publications



What I Did So Far

- Started with the goal of creating a search engine for components
- Analyzed different component standards
 - In particular, performance of WSs
- Component usage follows Web links patterns
 - This justified the usage of Web ranking algorithms
- Studied a possible solution based on P2P
 - Good for scalability and reliability
- Open problem was the collection selection



Document Partitioning and Collection Selection

- Very general problem with several applications
- Developed a joint strategy for the two tasks based on co-clustering
- Improved the performance of state-of-the-art selection (CORI) by creating good partitions
- Surpassed the state-of-the-art selection with a new strategy (PCAP)
- New more compact representation model



Towards a PhD Dissertation

- Implement an adaptive selection strategy:
 - with PCAP, one can estimate the number of clusters needed;
- complete a deeper analysis of the query-vector representation for IR tasks;
- compare document- and term-partitioning.



Related Publications

- Aldinucci M., Campa S., Coppola M., Danelutto M., Laforenza D., Puppini D., Scarponi L., Vanneschi M. Components for high-performance grid programming in Grid.IT "Component Models and Systems for Grid Applications", edited by Vladimir Getov and Thilo Kielmann, Springer, 2004
- Puppini D., Silvestri F. The Social Network of Java Classes SAC '06, April 23-27, 2006, Dijon, France.
- Puppini D., Silvestri F., Laforenza D. Query-Driven Document Partitioning and Collection Selection Infoscale 2006, May 30-June 1, 2006, Hong Kong.
- Puppini D., Moncelli S., Baraglia R., Tonello N., Silvestri F. A Grid Information Service Based on Peer-to-Peer in Proceedings of EuroPar2005. Springer LNCS 2648/2005.



Related Publications

- Silvestri F., Puppini D., Laforenza D., Orlando S. Toward a search engine for software components. IEEE Web Intelligence (Beijing, China, September 20-24, 2004). IEEE, 2004.
- Puppini D., Tonello N., Laforenza D. Using web services to run distributed numerical applications. 11th EuroPVM/MPI 2004 (Budapest, 19-22 settembre 2004). Springer LNCS 3241/2004
- Puppini D., Silvestri F., Laforenza D. An evaluation of component-based software design approaches. CCGRID 2004. 4th IEEE/ACM International Symposium on Cluster Computing and the Grid (Chicago, USA, 19-22 April, 2004).
- Puppini D., Tonello N., Laforenza D. How to Run Scientific Applications Over Web Services in Workshop on Web and Grid Services for Scientific Data Analysis (WAGSSDA), Oslo, Norway, June 2005



Related Publications

- Puppini D., Silvestri F., Laforenza D. Component Metadata Management and Publication for the Grid in Proceedings of the International Conference on Information Technology Coding and Computing ITCC 2005, Las Vegas, NV, USA, 4–6 April, 2005
- Diego Puppini, Matthieu Morel, Denis Caromel, Domenico Laforenza, Françoise Baude GRIDLE Search for the Fractal Component Model. To appear in Proceedings for the Pisa CoreGRID integration workshop, Pisa, November 2005, Springer



Courses/Seminar Series

- Summer School on Grid Computing
- Summer School on Mobile and Wireless Networks
- Semantic Foundations for Computer and Network Security (Gorrieri)
- Global Computing (Ferrari)
- (?) Operating Systems (Pelagatti)
- Bio-Informatics Seminar Series (Pisanti)
- Algorithms for Bio-Informatics (Zaki)
- Logica (Masini)
- Geometria Computazionale (Pellegrini)



Project Involvement

- FIRB Grid.it** Contributed to analyze and define a common component model, to develop a tool to restructure ASSIST applications
- CoreGrid** Involved in defining a common component model, discussed the possibility of searching components and their meta-data
- XtreemOS** Responsible for designing and implementing a scalable resource discovery system

