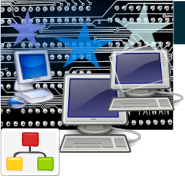


The design of BonjourGrid, a decentralized system for the coordination of multiple instances of Desktop Grid middleware

Christophe Cérin[1]
http://www.lipn.fr/~cerin/

[1]Université de Paris XIII, CNRS UMR 7030, France

APSCC'2010 invited paper

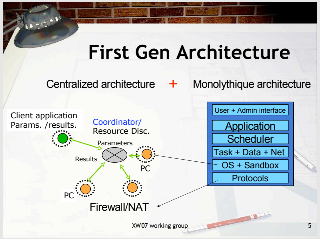
Table of contents

Objectives

1. Motivate research projets in Grids & Clouds. . . including a deep understanding of the eco-system for coordination;
2. Starting from recent advances in Desktop Grid Middleware:
   - BonjourGrid (orchestration of multiple instances of DG middleware) and PastryGrid (fully distributed execution of applications)
   - Joint works with UTIC lab., Tunisia (Leila Abidi, Heithem Abbes and Mohamed Jemni)
3. Before keeping innovative ideas to reuse. . .
   - for cloud proposal (ANR):
     - decentralized architectures and services; large scale systems (FT);
     - interoperability of services; service provisioning;
   - for competitiveness clusters in France (OSEO): the Resilience project.

Desktop Grid Architectures

Desktop Grid



First Gen Architecture

Centralized architecture + Monolythique architecture

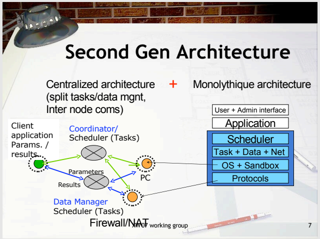
Client application
Params. /results.
Coordinator/
Resource Disc.
Parameters
Results
PC
PC
Firewall/NAT
User + Admin interface
Application
Scheduler
Task + Data + Net
OS + Sandbox
Protocols

XW07 working group 5

Key Points

- Federation of thousand of nodes;
- Internet as the communication layer: no trust!
- Volatility; local IP; Firewall

Desktop Grid



Second Gen Architecture

Centralized architecture (split tasks/data mgnt, Inter node coms) + Monolythique architecture

Future Generation (in 2006)

- Distributed Architecture
- Architecture with modularity: every component is "configurable": scheduler, storage, transport protocole
- Direct communications between peers;
- Security;
- Applications coming from any sciences (e-Science applications)

In search of distributed architecture

First line: publish/subscribe system to notify and coordinate services and multiple DG without a central broker ⇒ BonjourGrid;

Second line: approach based on structured overlay network to discover (on the fly) the next node executing the next task ⇒ PastryGrid;

(main contributions of Heithem Abbes in his PhD)

Main objectives of BonjourGrid

- Count on existing distributed tools for services discovering (publish/subscribe paradigm);

Main objectives of BonjourGrid

- Count on existing distributed tools for services discovering (publish/subscribe paradigm);
- Design and implement a platform able to manage multiple instances of DG middleware;

Main objectives of BonjourGrid

- Count on existing distributed tools for services discovering (publish/subscribe paradigm);
- Design and implement a platform able to manage multiple instances of DG middleware;
- Reduce as much as possible the use of any central element;

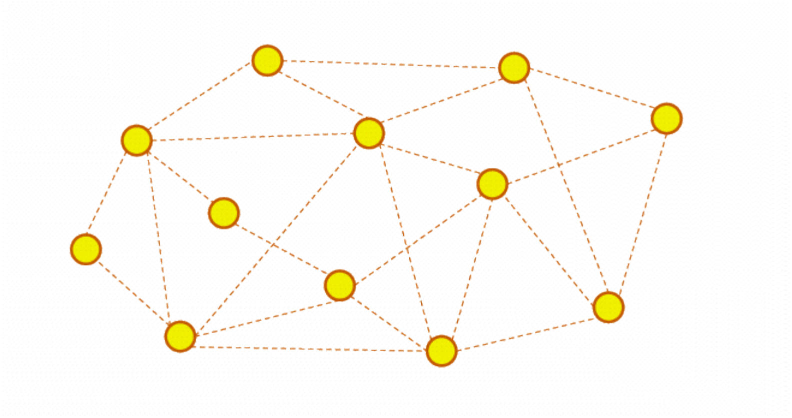
Main objectives of BonjourGrid

- Count on existing distributed tools for services discovering (publish/subscribe paradigm);
- Design and implement a platform able to manage multiple instances of DG middleware;
- Reduce as much as possible the use of any central element;
- Create a coordinator, on the fly, without any system administrator intervention; From a vision with a single coordinator towards a vision with multiple coordinators.

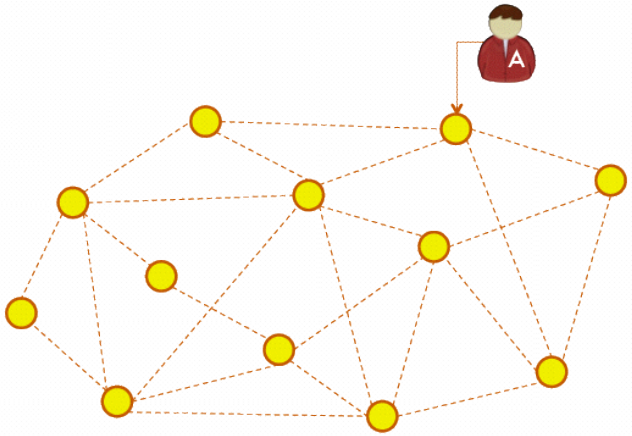
Main objectives of BonjourGrid

- Count on existing distributed tools for services discovering (publish/subscribe paradigm);
- Design and implement a platform able to manage multiple instances of DG middleware;
- Reduce as much as possible the use of any central element;
- Create a coordinator, on the fly, without any system administrator intervention; From a vision with a single coordinator towards a vision with multiple coordinators.
- Each coordinator searches, in a concurrent way, participants (idle machines)

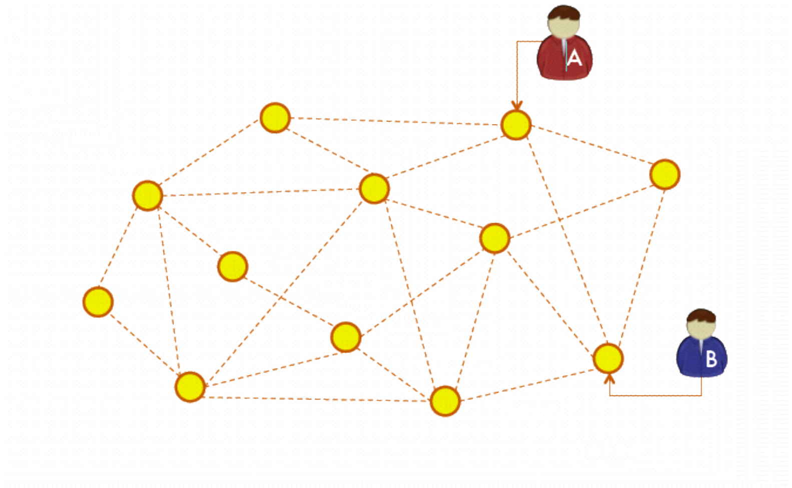
How BonjourGrid works



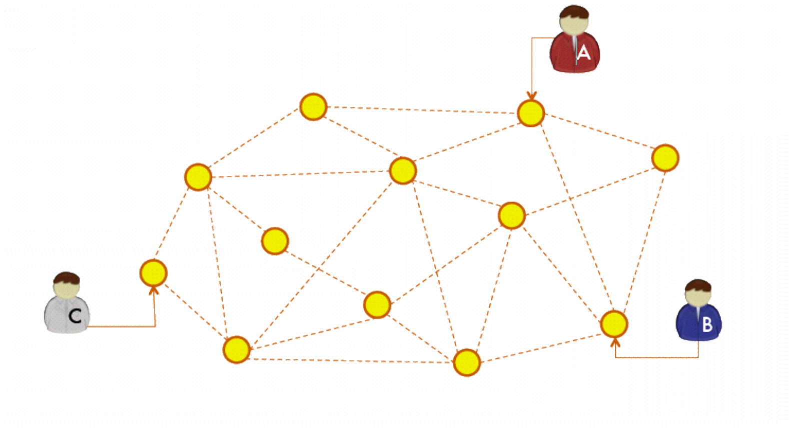
How BonjourGrid works



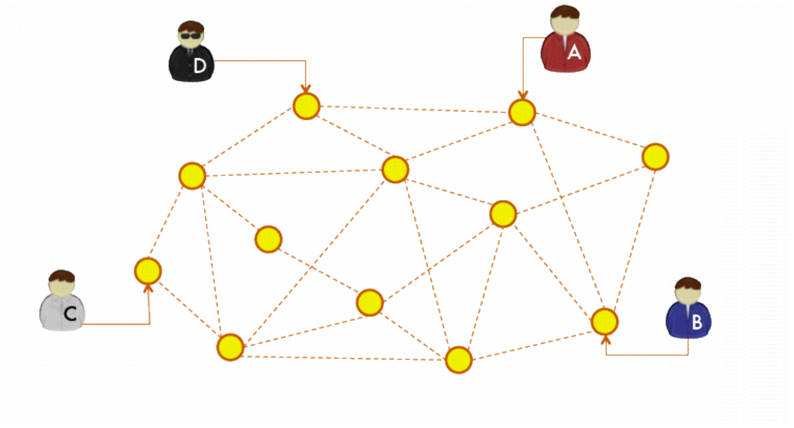
How BonjourGrid works



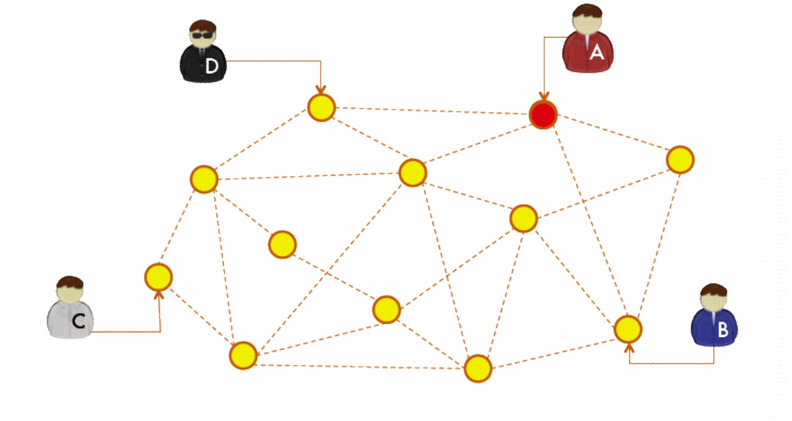
How BonjourGrid works



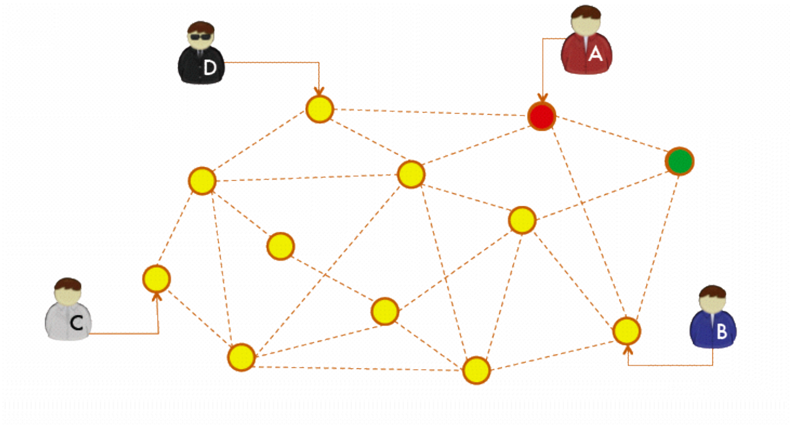
How BonjourGrid works



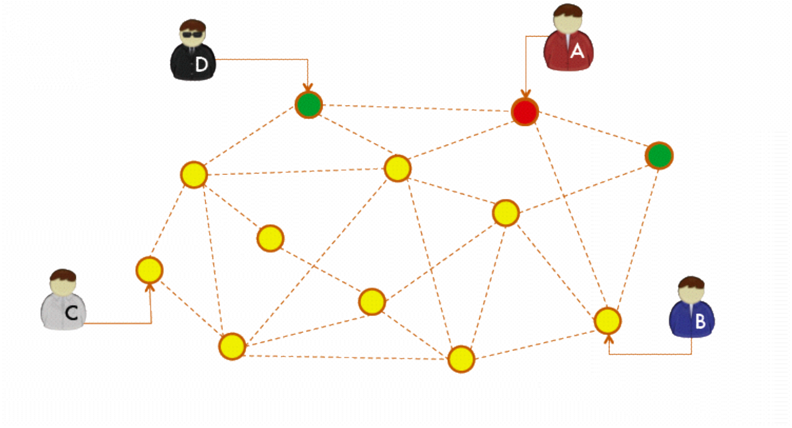
How BonjourGrid works



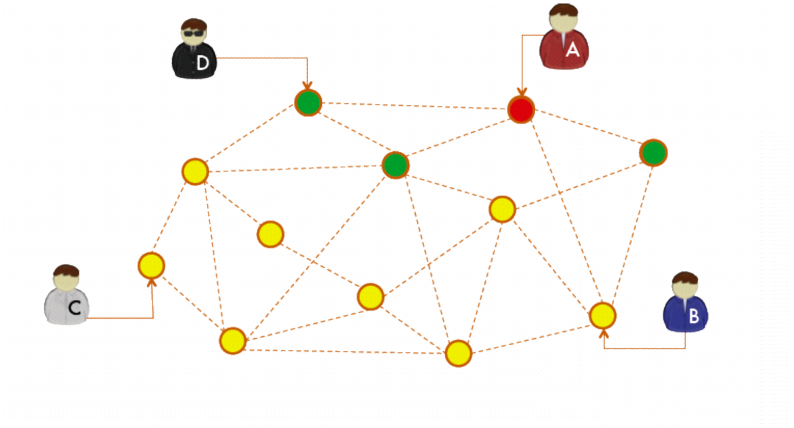
How BonjourGrid works



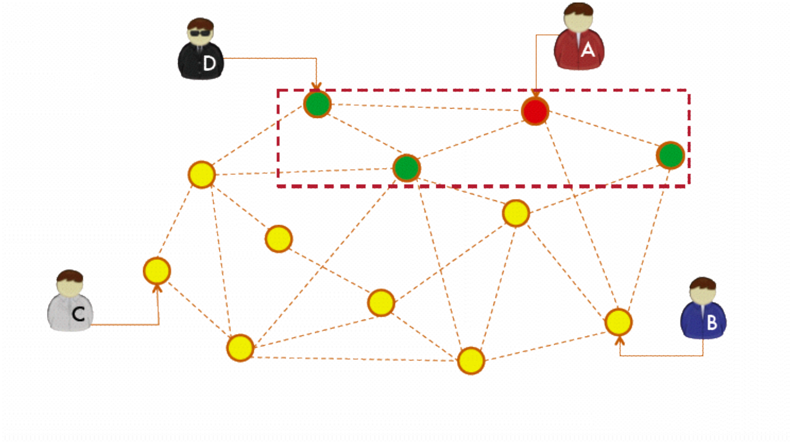
How BonjourGrid works



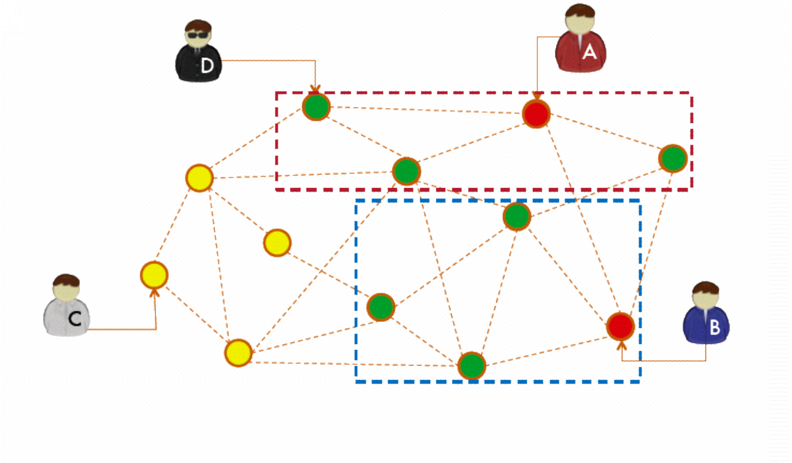
How BonjourGrid works



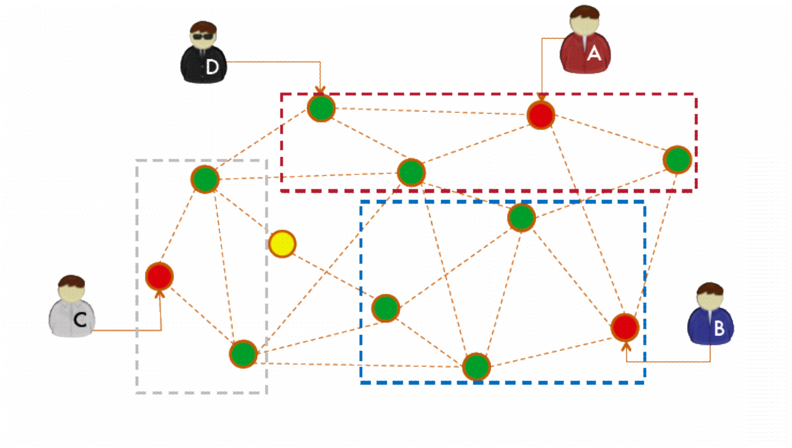
How BonjourGrid works



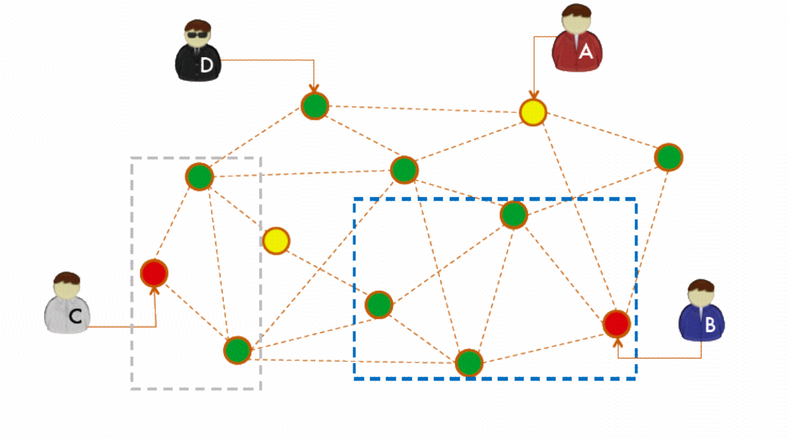
How BonjourGrid works



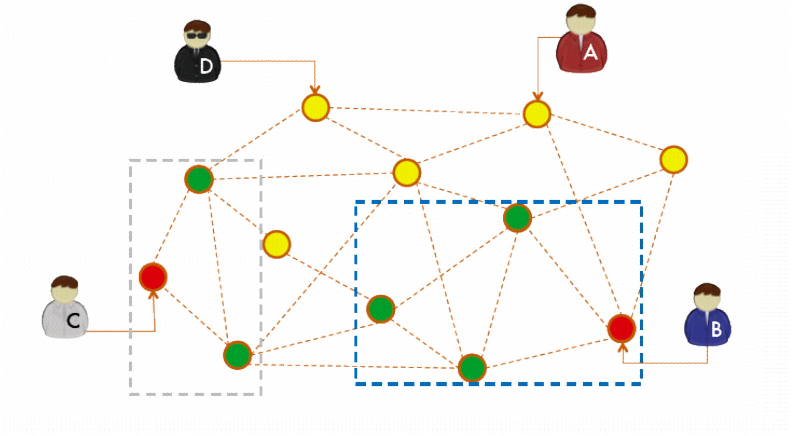
How BonjourGrid works



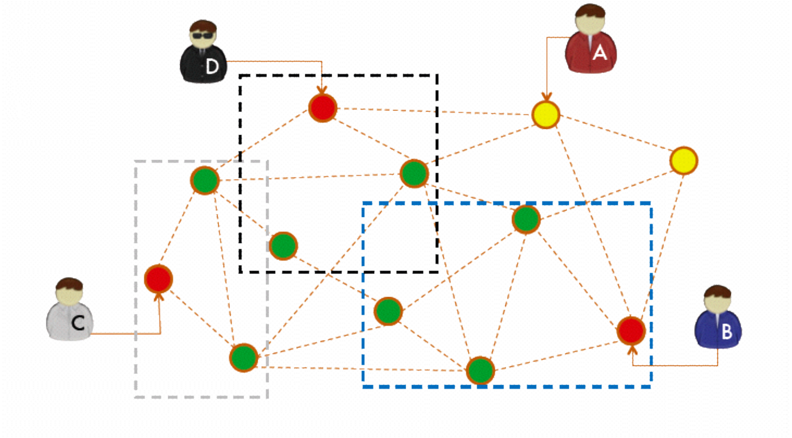
How BonjourGrid works



How BonjourGrid works



How BonjourGrid works



BonjourGrid vision

- The user requests for computation;

BonjourGrid vision

- The user requests for computation;
- The user provides the control flow graph, binaries, input data;

BonjourGrid vision

- The user requests for computation;
- The user provides the control flow graph, binaries, input data;
- The user deploys locally a coordinator and requests for participants; We support XtremWeb, Condor, Boinc.

BonjourGrid vision

- The user requests for computation;
- The user provides the control flow graph, binaries, input data;
- The user deploys locally a coordinator and requests for participants; We support XtremWeb, Condor, Boinc.
- The coordinator selects a set of machines (criteria: RAM, CPU, costs. . .)

BonjourGrid vision

- The user requests for computation;
- The user provides the control flow graph, binaries, input data;
- The user deploys locally a coordinator and requests for participants; We support XtremWeb, Condor, Boinc.
- The coordinator selects a set of machines (criteria: RAM, CPU, costs. . .)
- Upon completion, the coordinator returns to the idle state, slaves are freed and the coordination protocol:

BonjourGrid vision

- The user requests for computation;
- The user provides the control flow graph, binaries, input data;
- The user deploys locally a coordinator and requests for participants; We support XtremWeb, Condor, Boinc.
- The coordinator selects a set of machines (criteria: RAM, CPU, costs. . .)
- Upon completion, the coordinator returns to the idle state, slaves are freed and the coordination protocol:
  - manages and controls resources, services and computing elements;
  - does not depend on any specific machine nor any central element.

BonjourGrid vision

- The user requests for computation;
- The user provides the control flow graph, binaries, input data;
- The user deploys locally a coordinator and requests for participants; We support XtremWeb, Condor, Boinc.
- The coordinator selects a set of machines (criteria: RAM, CPU, costs. . .)
- Upon completion, the coordinator returns to the idle state, slaves are freed and the coordination protocol:
  - manages and controls resources, services and computing elements;
  - does not depend on any specific machine nor any central element.

The protocol for resources discovering

- Based on Bonjour from Apple;

The protocol for resources discovering

- Based on Bonjour from Apple;
- A publish/subscribe system is easy to use (toolbox = publish(), subscribe(), browse())

The protocol for resources discovering

- Based on Bonjour from Apple;
- A publish/subscribe system is easy to use (toolbox = publish(), subscribe(), browse())
- Bonjour is dedicated to LAN (wide area Bonjour? We need some experiments)

The protocol for resources discovering

- Based on Bonjour from Apple;
- A publish/subscribe system is easy to use (toolbox = publish(), subscribe(), browse())
- Bonjour is dedicated to LAN (wide area Bonjour? We need some experiments)
- Concerning the Wide Area implementation: we can also think about Apache Kandula (http://ws.apache.org/kandula/) or even Cisco Jabber protocol (http://www.jabber.com):

The protocol for resources discovering

- Based on Bonjour from Apple;
- A publish/subscribe system is easy to use (toolbox = publish(), subscribe(), browse())
- Bonjour is dedicated to LAN (wide area Bonjour? We need some experiments)
- Concerning the Wide Area implementation: we can also think about Apache Kandula (http://ws.apache.org/kandula/) or even Cisco Jabber protocol (http://www.jabber.com):
  - Extensible Messaging and Presence Protocol (XMPP) (formerly named Jabber) is an open, XML-based protocol originally aimed at near-real-time, extensible instant messaging (IM) and presence information, but now expanded into the broader realm of message-oriented middleware

The protocol for resources discovering

- Based on Bonjour from Apple;
- A publish/subscribe system is easy to use (toolbox = publish(), subscribe(), browse())
- Bonjour is dedicated to LAN (wide area Bonjour? We need some experiments)
- Concerning the Wide Area implementation: we can also think about Apache Kandula (http://ws.apache.org/kandula/) or even Cisco Jabber protocol (http://www.jabber.com):
  - Extensible Messaging and Presence Protocol (XMPP) (formerly named Jabber) is an open, XML-based protocol originally aimed at near-real-time, extensible instant messaging (IM) and presence information, but now expanded into the broader realm of message-oriented middleware
- The current protocol has been developed/specified with 'ad-hoc' methods → we need to consolidate the trust (ongoing project to verify it, based on Colored Petri Nets)

Fault Tolerance with BonjourGrid

- Intrinsic property of any large scale system;

Fault Tolerance with BonjourGrid

- Intrinsic property of any large scale system;
- We assume that any coordinator is responsible for its FT (it manages the volatility of attached slaves)

Fault Tolerance with BonjourGrid

- Intrinsic property of any large scale system;
- We assume that any coordinator is responsible for its FT (it manages the volatility of attached slaves)
- Our solution: tolerate the failure of coordinators

Fault Tolerance with BonjourGrid

- Intrinsic property of any large scale system;
- We assume that any coordinator is responsible for its FT (it manages the volatility of attached slaves)
- Our solution: tolerate the failure of coordinators
  - For any application we create and manage dynamically copies of the coordinator;
  - We manage k copies; based on passive replication.
  - When a service disappears: we added a special status flag to distinguish between 'end of the application' / 'failure' $\Rightarrow$ slaves can redirect the communication to a copy.

Intensive Experiments

- BonjourGrid has been tested intensively: stressed scenario to more relaxing scenario

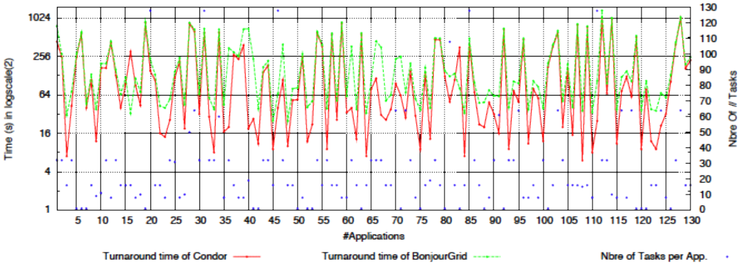
Intensive Experiments

- BonjourGrid has been tested intensively: stressed scenario to more relaxing scenario
  - in terms of #coordinator versus #nodes
  - in terms of using virtual machines to reach 1000 nodes;
  - in terms of comparing Boinc, Condor, XtremWeb over our protocol;
  - in terms of robustness in supporting FT;

Intensive Experiments

- BonjourGrid has been tested intensively: stressed scenario to more relaxing scenario
  - in terms of #coordinator versus #nodes
  - in terms of using virtual machines to reach 1000 nodes;
  - in terms of comparing Boinc, Condor, XtremWeb over our protocol;
  - in terms of robustness in supporting FT;
- Example Condor: 130 applications (2 to 128 // tasks), 200 nodes, application task: 1s to 500s. Result: with BonjourGrid, 35% of applications generate a delay of about 30s.

Experiments: one example



Motivations for formal verification

Concerns about BoujourGrid

- Built with ad-hoc methods;
- Get more confidence into the protocol; (ex. are you sure that N alive workers $\Rightarrow$ at least one alive coordinator?)
- Refine it to add new properties (ex. how we manage/become a coordinator ; how we cancel a coordinator ; control how the workers leave the system)

Motivations for formal verification

Concerns about BoujourGrid

- Built with ad-hoc methods;
- Get more confidence into the protocol; (ex. are you sure that N alive workers $\Rightarrow$ at least one alive coordinator?)
- Refine it to add new properties (ex. how we manage/become a coordinator ; how we cancel a coordinator ; control how the workers leave the system)

Concerns about Pub/Sub

- The quest is to modelize any Pub/Sub protocol (or a system developed according to Pub/Sub paradigm);

Motivations for formal verification

Concerns about BoujourGrid

- Built with ad-hoc methods;
- Get more confidence into the protocol; (ex. are you sure that N alive workers $\Rightarrow$ at least one alive coordinator?)
- Refine it to add new properties (ex. how we manage/become a coordinator ; how we cancel a coordinator ; control how the workers leave the system)

Concerns about Pub/Sub

- The quest is to modelize any Pub/Sub protocol (or a system developed according to Pub/Sub paradigm);
- BonjourGrid is just a 'case study';

Motivations for formal verification

Concerns about BoujourGrid

- Built with ad-hoc methods;
- Get more confidence into the protocol; (ex. are you sure that N alive workers $\Rightarrow$ at least one alive coordinator?)
- Refine it to add new properties (ex. how we manage/become a coordinator ; how we cancel a coordinator ; control how the workers leave the system)

Concerns about Pub/Sub

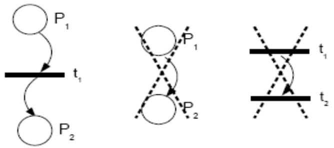
- The quest is to modelize any Pub/Sub protocol (or a system developed according to Pub/Sub paradigm);
- BonjourGrid is just a 'case study';
- See `http://event-based.org/` for efforts in modelization and verification of event-based systems

Tools

- Petri Nets is of premier choice for the specification and verification of distributed systems (assertion verified by experience);
- Our Lab has experts in the field; (we are coming from the HPC community);
- Advantages: graphical tool; state exploration can be made; other 'specification language' can generate a Petri Net representation.

Petri Net howto

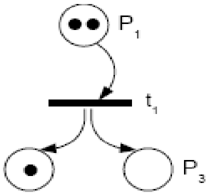
A Petri net is a *bipartite oriented graph* with two types of vertexes.



Arcs represent 'events' and vertexes represent states or conditions to satisfy before the execution of some 'event'.

Petri Net howto

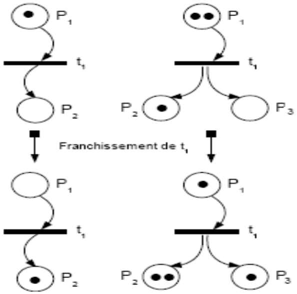
Each vertex contains one or more marking (token). The marking defines the state of the network and thu the state of the system.



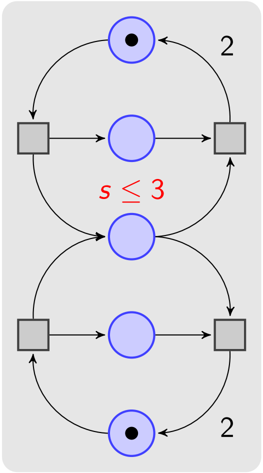
Petri Net howto

To simulate the dynamic behavior (to pass from one state to another one): state crossing
Rules for crossing:

- crossing is an atomic operation;
- a token is consummated in every 'input' vertex;
- a token is produced in each 'output' vertex;

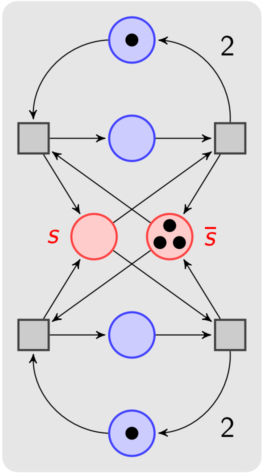


Petri net design choice



replacement of the capacity by two places

$\rightsquigarrow$



Colored Petri Net (CPN)

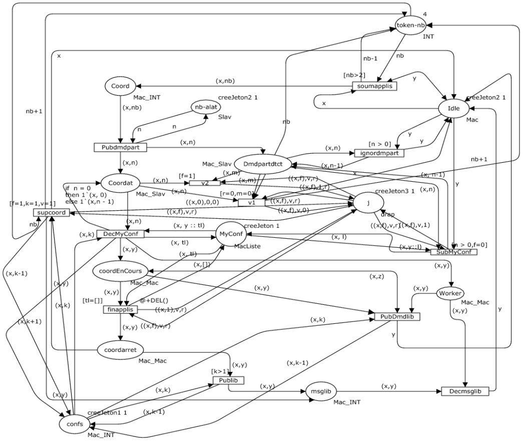
- colors serve to distinguish between tokens (events);
- colors to modelize different elements in the same state;
- colors to reduce the size of the Petri net.

Colored Petri Net (CPN)

- colors serve to distinguish between tokens (events);
- colors to modelize different elements in the same state;
- colors to reduce the size of the Petri net.
- a tool exists: CPNtools (`http://cpntools.org/start`)

Initial Petri net model for BonjourGrid

The 3 states protocol (Idle, Coordinator, Worker) of BonjourGrid has been modelized with a PN of 13 vertexes and. . . too many arcs.



Initial Petri net model for BonjourGrid

Criticism – – –

- Difficult to isolate what is specific to the Pub/Sub paradigm and what is related to BonjourGrid;
- The building of PN has been made by an ad-hoc method (at some level it is always true: formal methods exist but the idea of the proof / specification is an informal way of thinking)

Initial Petri net model for BonjourGrid

Criticism − − −

- Difficult to isolate what is specific to the Pub/Sub paradigm and what is related to BonjourGrid;
- The building of PN has been made by an ad-hoc method (at some level it is always true: formal methods exist but the idea of the proof / specification is an informal way of thinking)

Criticism + + +

- Details / choices have been introduced in the model;
- We can use the facility of CPNtool for proof carrying.

Proofs conducted on the model

CPNtool generates the 'state space' of the system (all the possible markings):

- Good news: no deadlock (the initial marking is the home marking) ; no dead marking (we can leave any marking) ; reachability (a marking M' is reachable from another marking M) ; liveness (there is a single dead marking) ; boundedness (how many and which tokens a place may hold, when all reachable markings are considered?).
- Bad news: a worker may exists. . . while there is no coordinator in the 'Coord' state ; the last worker cannot be canceled before its coordinator.

Initial Petri net model: partial conclusion

- Good news: the feedback received from our modelization has allowed a better understanding of the *nature* of the BonjourGrid protocol;
- Bad news: the modelization do to *capture* or *isolate* the deep mechanisms of any Pub/Sub system;

BonjourGrid design revisited

- We are currently specifying the Pub/Sub paradigm according to Petri Net and UML state transition diagram;
- See also: Modeling publish/subscribe systems based on colored Petri nets ZHU Lian-zhang, LIU Fan (College of Computer and Communication Engineering, China University of Petroleum, Dongying 257061, China)

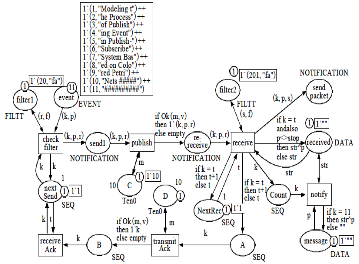


图 2　事件发布的 CPN 模型

BonjourGrid design revisited

- Properties of the design: (a) separation between Pub/Sub and what is relevant / specific to BonjourGrid (b) allow the *composition* of the Pub/Sub *substrate* with the dedicated protocol

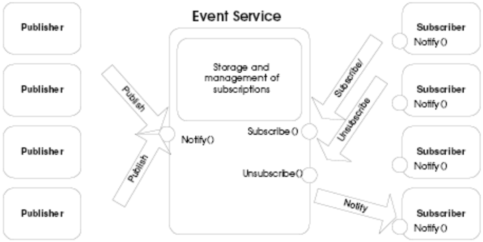
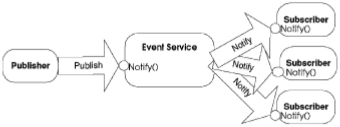
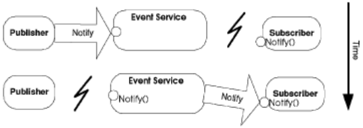


Figure: Pub/Sub by A.M. Kermarrec and al.

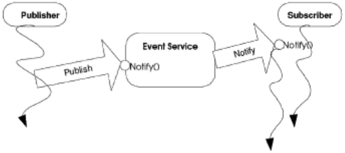
Properties of the Pub/Sub paradigm



Space decoupling



Time decoupling



Synchronization decoupling

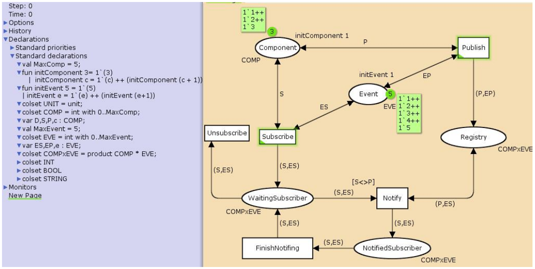
So good for the scalability of your system. . . better than RPC!

Properties of the Pub/Sub paradigm

- Space decoupling: the interacting entities do not need to know each other;
- Time decoupling: the interacting entities do not need to be actively participating in the interaction at the same time;
- Synchronization decoupling: publishers are not blocked while producing events, and subscribers can get asynchronously notified (through a callback) of the occurrence of an event while performing some concurrent activity.

The Petri Net model for Pub/Sub (ongoing work)

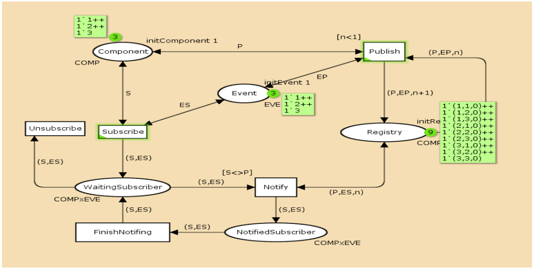
Remind that Petri Net is good for reasoning about your protocol.



```
Step: 0
Time: 0
▶Options
▶History
▼Declarations
  ▶Standard priorities
  ▼Standard declarations
    ▼val MaxComp = 5;
    ▼fun initComponent 3= 1`(3)
        |  initComponent c = 1`(c) ++ (initComponent (c + 1))
    ▼fun initEvent 5 = 1`(5)
      | initEvent e = 1`(e) ++ (initEvent (e+1))
    ▼colset UNIT = unit;
    ▼colset COMP = int with 0..MaxComp;
    ▼var D,S,P,c : COMP;
    ▼val MaxEvent = 5;
    ▼colset EVE = int with 0..MaxEvent;
    ▼var ES,EP,e : EVE;
    ▼colset COMPxEVE = product COMP * EVE;
    ▶colset INT
    ▶colset BOOL
    ▶colset STRING
▶Monitors
 New Page
```

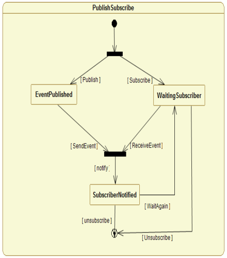
The Petri Net model for Pub/Sub (ongoing work)

Again, remind that Petri Net is good for reasoning about your protocol.



The UML model for Pub/Sub (ongoing work)

- UML is a widespread modeling language used in both industry and academia despite of its informal semantics and of some ambiguities;
- We count on a tool (developed at Paris XIII) which automatically translate 'UML' state diagrams to CPN. See "Formal Verification of UML State Diagrams: a Petri net based approach by Christine Choppy, Kais Klai and Hacene Zidani", UML&FM 2010.



events)

Partial conclusion about the modelization

- Current work: define the composition i.e. an inter-logic that matches two transitions from different models by connecting them via a place (and corresponding arcs);
- Ultimate Goal: automatically generate XMPP code (or whatever else with Pub/Sub support) from the specification. . . and the code is also proved!
- A cross discipline approach (people from HPC/GRID and people from Formal Methods);

<blink>Request for participation</blink>

Partial conclusion about the modelization

- Current work: define the composition i.e. an inter-logic that matches two transitions from different models by connecting them via a place (and corresponding arcs);
- Ultimate Goal: automatically generate XMPP code (or whatever else with Pub/Sub support) from the specification. . . and the code is also proved!
- A cross discipline approach (people from HPC/GRID and people from Formal Methods);

<blink>Request for participation</blink>

<blink><blink>Open Forum for Coordination and Provisioning in Clouds (OFCPC)</blink></blink>

Towards PaaS and Clouds

The new context: Platform as a Service and Cloud

- Outsourcing of software resources (Google word/spreadsheet online) and hardware resources (Amazon EC2);

Towards PaaS and Clouds

The new context: Platform as a Service and Cloud

- Outsourcing of software resources (Google word/spreadsheet online) and hardware resources (Amazon EC2);
- A variant: Platform as a Service (PaaS) where users also inherit from a complete development infrastructure (based on Javascript for the future?);

Towards PaaS and Clouds

The new context: Platform as a Service and Cloud

- Outsourcing of software resources (Google word/spreadsheet online) and hardware resources (Amazon EC2);
- A variant: Platform as a Service (PaaS) where users also inherit from a complete development infrastructure (based on Javascript for the future?);
  - No hosting problem for the user;

Towards PaaS and Clouds

The new context: Platform as a Service and Cloud

- Outsourcing of software resources (Google word/spreadsheet online) and hardware resources (Amazon EC2);
- A variant: Platform as a Service (PaaS) where users also inherit from a complete development infrastructure (based on Javascript for the future?);
  - No hosting problem for the user;
  - No update problem for the user (he always uses the last release);

Towards PaaS and Clouds

The new context: Platform as a Service and Cloud

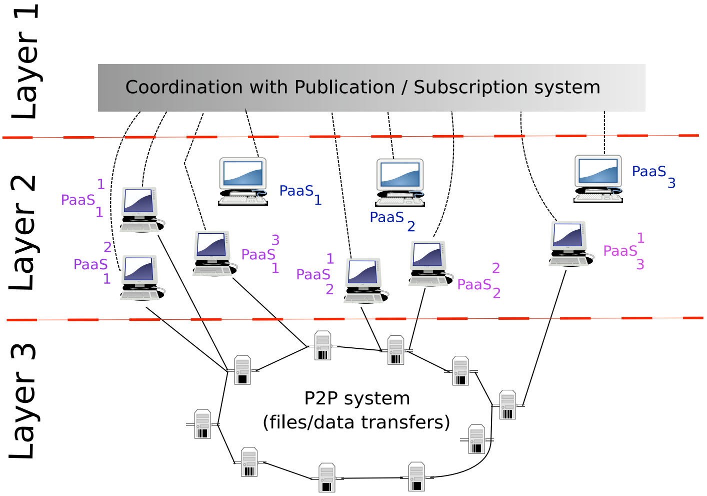
- Outsourcing of software resources (Google word/spreadsheet online) and hardware resources (Amazon EC2);
- A variant: Platform as a Service (PaaS) where users also inherit from a complete development infrastructure (based on Javascript for the future?);
  - No hosting problem for the user;
  - No update problem for the user (he always uses the last release);
  - No maintenance, no local storage.

Towards PaaS and Clouds

The new context: Platform as a Service and Cloud

- Outsourcing of software resources (Google word/spreadsheet online) and hardware resources (Amazon EC2);
- A variant: Platform as a Service (PaaS) where users also inherit from a complete development infrastructure (based on Javascript for the future?);
  - No hosting problem for the user;
  - No update problem for the user (he always uses the last release);
  - No maintenance, no local storage.
- We have started an initiative for defining and designing a "general purpose PaaS" based on distributed protocols for coordination and data exchange.

Architecture overview of the research project



Key points regarding Philosophy

- Different instances (say, of a database) want to exchange data temporally $\Rightarrow$ an open protocol does not capture the user

Key points regarding Philosophy

- Different instances (say, of a database) want to exchange data temporally ⇒ an open protocol does not capture the user
- Different instances (say, of an ERP) do public announcements to search for providers, then explore HTML links (interrogate different Clouds able to answer) ⇒ an open protocol do not capture the user, again

Key points regarding Philosophy

- Different instances (say, of a database) want to exchange data temporally $\Rightarrow$ an open protocol does not capture the user
- Different instances (say, of an ERP) do public announcements to search for providers, then explore HTML links (interrogate different Clouds able to answer) $\Rightarrow$ an open protocol do not capture the user, again
- So, we want open protocols to coordinate and to exchange data!

Key points regarding Philosophy

- Different instances (say, of a database) want to exchange data temporally ⇒ an open protocol does not capture the user
- Different instances (say, of an ERP) do public announcements to search for providers, then explore HTML links (interrogate different Clouds able to answer) ⇒ an open protocol do not capture the user, again
- So, we want open protocols to coordinate and to exchange data!

Some Challenges

- Where to insert the different connectors in the PaaS software stack to get an open infrastructure?

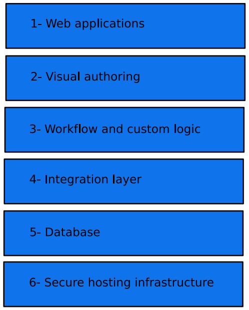


Figure: Software stack in PaaS (source: Coghead)

Research opportunities

Above the Clouds: A Berkeley View of Cloud Computing

- Availability of a service $\rightarrow$ mastering FT $\rightarrow$ redundancy;

Research opportunities

Above the Clouds: A Berkeley View of Cloud Computing

- Availability of a service $\rightarrow$ mastering FT $\rightarrow$ redundancy;
- Data Lock-In: API must not be proprietary but should rely on open standards;

Research opportunities

Above the Clouds: A Berkeley View of Cloud Computing

- Availability of a service → mastering FT → redundancy;
- Data Lock-In: API must not be proprietary but should rely on open standards;
- Data Transfer Bottlenecks → use P2P techniques;

Research opportunities

Above the Clouds: A Berkeley View of Cloud Computing

- Availability of a service $\rightarrow$ mastering FT $\rightarrow$ redundancy;
- Data Lock-In: API must not be proprietary but should rely on open standards;
- Data Transfer Bottlenecks $\rightarrow$ use P2P techniques;
- Bugs in Large-Scale Distributed Systems $\rightarrow$ use Formal Methods to specify and to analyse architecture and protocols;

Research opportunities

Above the Clouds: A Berkeley View of Cloud Computing

- Availability of a service → mastering FT → redundancy;
- Data Lock-In: API must not be proprietary but should rely on open standards;
- Data Transfer Bottlenecks → use P2P techniques;
- Bugs in Large-Scale Distributed Systems → use Formal Methods to specify and to analyse architecture and protocols;
- Scaling Quickly → instanciate new PaaS on the fly → define a model for cooperation and interaction;

Research opportunities

Above the Clouds: A Berkeley View of Cloud Computing

- Availability of a service $\rightarrow$ mastering FT $\rightarrow$ redundancy;
- Data Lock-In: API must not be proprietary but should rely on open standards;
- Data Transfer Bottlenecks $\rightarrow$ use P2P techniques;
- Bugs in Large-Scale Distributed Systems $\rightarrow$ use Formal Methods to specify and to analyse architecture and protocols;
- Scaling Quickly $\rightarrow$ instanciate new PaaS on the fly $\rightarrow$ define a model for cooperation and interaction;

State of the Art

Similar projects

- Vertebra (http://www.engineyard.com/): "Service-Oriented-Architecture for the cloud, an application deployment platform" – based on Ruby and Erlang. The project moves to "On-demand deployment and management of your Ruby on Rails applications with Engine Yard Cloud – One-click code deploys, application cloning, data automation. . ."

State of the Art

Similar projects

- Vertebra (http://www.engineyard.com/): "Service-Oriented-Architecture for the cloud, an application deployment platform" – based on Ruby and Erlang. The project moves to "On-demand deployment and management of your Ruby on Rails applications with Engine Yard Cloud – One-click code deploys, application cloning, data automation. . ."
- Kandula (http://ws.apache.org/kandula/): Presently Kandula implements WS-Coordination and WS-AtomicTransaction protocols. WS-BusinessActivity protocol will be available in the near future.
  Kandula project has 2 branches. Kandula1 branch runs on Apache Axis 1.x. Kandula2 branch (new) runs on Apache Axis2

State of the Art

Similar projects

- Vertebra (http://www.engineyard.com/): "Service-Oriented-Architecture for the cloud, an application deployment platform" – based on Ruby and Erlang. The project moves to "On-demand deployment and management of your Ruby on Rails applications with Engine Yard Cloud – One-click code deploys, application cloning, data automation..."
- Kandula (http://ws.apache.org/kandula/): Presently Kandula implements WS-Coordination and WS-AtomicTransaction protocols. WS-BusinessActivity protocol will be available in the near future.
  Kandula project has 2 branches. Kandula1 branch runs on Apache Axis 1.x. Kandula2 branch (new) runs on Apache Axis2

State of the Art

Similar projects

- Axis (http://ws.apache.org/axis2/): Axis2 is a Web Services / SOAP / WSDL engine, the successor to the widely used Apache Axis SOAP stack. There are two implementations of the Apache Axis2 Web services engine - Apache Axis2/Java and Apache Axis2/C

State of the Art

Similar projects

- Axis (http://ws.apache.org/axis2/): Axis2 is a Web Services / SOAP / WSDL engine, the successor to the widely used Apache Axis SOAP stack. There are two implementations of the Apache Axis2 Web services engine - Apache Axis2/Java and Apache Axis2/C
- Corona: A High Performance Publish-Subscribe System for the World (http://www.truststc.org/pubs/38.html). Topic based publish subscribe system for the Web. URLs of Web content serve as topics or channels. Any Web object identifiable by an URL can be monitored with Corona

State of the Art

Similar projects

- Axis (http://ws.apache.org/axis2/): Axis2 is a Web Services / SOAP / WSDL engine, the successor to the widely used Apache Axis SOAP stack. There are two implementations of the Apache Axis2 Web services engine - Apache Axis2/Java and Apache Axis2/C
- Corona: A High Performance Publish-Subscribe System for the World (http://www.truststc.org/pubs/38.html). Topic based publish subscribe system for the Web. URLs of Web content serve as topics or channels. Any Web object identifiable by an URL can be monitored with Corona
- UDDI, XMMP...

State of the Art

Similar projects

- Axis (http://ws.apache.org/axis2/): Axis2 is a Web Services / SOAP / WSDL engine, the successor to the widely used Apache Axis SOAP stack. There are two implementations of the Apache Axis2 Web services engine - Apache Axis2/Java and Apache Axis2/C
- Corona: A High Performance Publish-Subscribe System for the World (http://www.truststc.org/pubs/38.html). Topic based publish subscribe system for the Web. URLs of Web content serve as topics or channels. Any Web object identifiable by an URL can be monitored with Corona
- UDDI, XMPP...

State of the Art

Pieces of the maze (not exhaustive)

- Data exchange: Bitdew (Gilles Fedak from INRIA). Have also a look to SyncML Protocol (http://www.openmobilealliance.org/syncml/): This open standard seeks to drive data mobility by establishing a common language for communications among devices, applications, and networks.

State of the Art

Pieces of the maze (not exhaustive)

- Data exchange: Bitdew (Gilles Fedak from INRIA). Have also a look to SyncML Protocol (http://www.openmobilealliance.org/syncml/): This open standard seeks to drive data mobility by establishing a common language for communications among devices, applications, and networks.
- TioLive (http://www.tiolive.com): Open source by Nexedi Corp. for Communication: email, telephone, chat, Backoffice: contacts, documents, accounting, ERP, CRM, e-Business: web site, e-Commerce:

State of the Art

Pieces of the maze (not exhaustive)

- Data exchange: Bitdew (Gilles Fedak from INRIA). Have also a look to SyncML Protocol (http://www.openmobilealliance.org/syncml/): This open standard seeks to drive data mobility by establishing a common language for communications among devices, applications, and networks.
- TioLive (http://www.tiolive.com): Open source by Nexedi Corp. for Communication: email, telephone, chat, Backoffice: contacts, documents, accounting, ERP, CRM, e-Business: web site, e-Commerce:

State of the Art

Pieces of the maze (not exhaustive)

- Programming: https://www.myerp5.com/kb/developer-How.To.Become.ERP5.Developer/view
- OSOE course: http://www.osoe-project.org/lesson
- TioLive tutorial: https://www.tiolive.com/documentation/tiolive-tutorial
- Documentation for developers:
  https://www.myerp5.com/kb/documentation_section/developer/
  https://www.myerp5.com/kb/documentation_section/developer/developer-Technology/view
  https://www.myerp5.com/kb/documentation_section/developer/
  enterprise-High..Performance.Zope/view

The Resilience project

SYSTEMATIC Groupe Thématique Logiciel Libre

- In France, a competitiveness cluster is an initiative that brings together companies, research centers and educational institutions in order to develop synergies and cooperative efforts.
- In Paris region: `http://www.systematic-paris-region.org/`
- Inside System@tic: open source initiative ⇒ Resilience (Nexedi, Nuxeo, Morpho, Gontran, ViFiB, Wallix, Xwiki, Alixen, Alterway, TCA, Institut Télécom, INRIA, Université Paris XIII)
- Resilience: Small and Medium size Entreprise & Research Institutes. 24 months; Cost: 4,353 K€
- French competitiveness clusters ↔ chinese science parks: `http://www.ambafrance-cn.org/Coopol-program.html`

The Resilience project

- Resilience: to resist damage and recover quickly from disturbances;
- Goal: promote and complement french initiatives in Cloud Computing (http://www.freecloudalliance.org/);
- Focus: resilience and data protection. The project reduces the risk of business intelligence or data loss associated with the increasing use of Cloud

$$\downarrow$$

if Google Calendar is used by the French embassy for Science and Technology in Jxxxx $\rightarrow$ NSA in US knows the agenda (Patriot Act)!

The Resilience sub-projects

- SafeDOC: how to store, cyphering of `.doc` that you edit with your navigator inside a Javascript technology? Google Docs is replaced by UNG (`http://www.freecloudalliance.org/ung-Home.Page`): Web Office and Web Groupware with javascript technology;

The Resilience sub-projects

- SafeDOC: how to store, cyphering of `.doc` that you edit with your navigator inside a Javascript technology? Google Docs is replaced by UNG (`http://www.freecloudalliance.org/ung-Home.Page`): Web Office and Web Groupware with javascript technology;
- JIO (Javascript I/O): redundancy, splitting, tattooing and encryption;

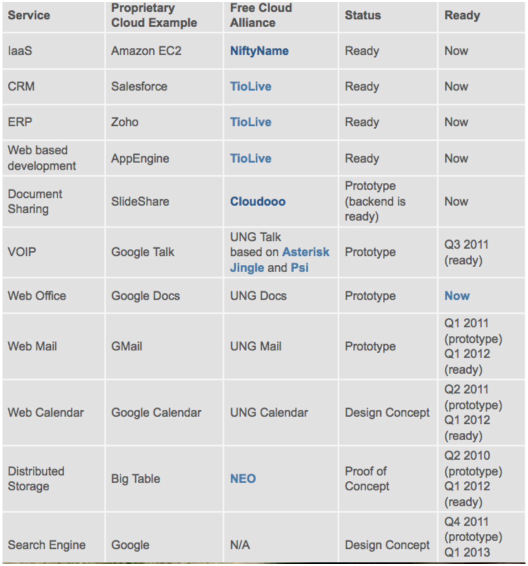
The Resilience sub-projects

- SafeDOC: how to store, cyphering of `.doc` that you edit with your navigator inside a Javascript technology? Google Docs is replaced by UNG (`http://www.freecloudalliance.org/ung-Home.Page`): Web Office and Web Groupware with javascript technology;
- JIO (Javascript I/O): redundancy, splitting, tattooing and encryption;
- SafeDYN: Javascript process in isolation;

The Resilience sub-projects

- SafeDOC: how to store, cyphering of `.doc` that you edit with your navigator inside a Javascript technology? Google Docs is replaced by UNG (`http://www.freecloudalliance.org/ung-Home.Page`): Web Office and Web Groupware with javascript technology;
- JIO (Javascript I/O): redundancy, splitting, tattooing and encryption;
- SafeDYN: Javascript process in isolation;

UNG: an universal framework to data access in Javascript

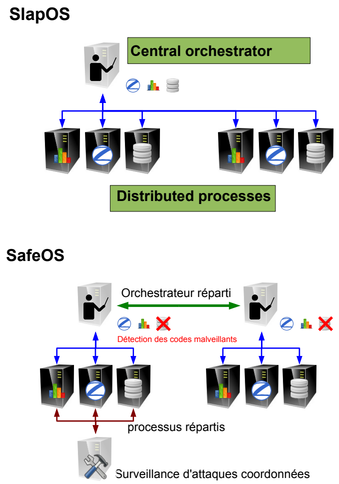


Service	Proprietary Cloud Example	Free Cloud Alliance	Status	Ready
IaaS	Amazon EC2	NiftyName	Ready	Now
CRM	Salesforce	TioLive	Ready	Now
ERP	Zoho	TioLive	Ready	Now
Web based development	AppEngine	TioLive	Ready	Now
Document Sharing	SlideShare	Cloudooo	Prototype (backend is ready)	Now
VOIP	Google Talk	UNG Talk based on Asterisk Jingle and Psi	Prototype	Q3 2011 (ready)
Web Office	Google Docs	UNG Docs	Prototype	Now
Web Mail	GMail	UNG Mail	Prototype	Q1 2011 (prototype) Q1 2012 (ready)
Web Calendar	Google Calendar	UNG Calendar	Design Concept	Q2 2011 (prototype) Q1 2012 (ready)
Distributed Storage	Big Table	NEO	Proof of Concept	Q2 2010 (prototype) Q1 2012 (ready)
Search Engine	Google	N/A	Design Concept	Q4 2011 (prototype) Q1 2013

The Resilience sub-projects

- based on SlapOS: processes orchestration + compensation and virtual currency management + a generic instantiation of application configurations + secure execution and intrusion detection;
  - Paris XIII: replace the centralized orchestrator by a decentralized one (and based on the idea of BonjourGrid)
  - Orchestrator = directory services + catalog of software + system for resource reservation;
  - Issues: a node is not overloaded; delegation of trust between entities; accounting and compensation; only certified codes

SafeOS: before and after



Conclusion

Hope

- DG has proved to be relevant for resource sharing ⇒ transpose this success story to the Cloud and PaaS universes ⇒ offer a technical alternate to Google, Salesforce, Amazon big farm of servers

Conclusion

Hope

- DG has proved to be relevant for resource sharing $\Rightarrow$ transpose this success story to the Cloud and PaaS universes $\Rightarrow$ offer a technical alternate to Google, Salesforce, Amazon big farm of servers
- Our approaches are based on emerging open source Cloud solution. From an economic point of view: if it is less expensive to host services locally and if it offers more advantages (we are not “dependent on a technology” $\rightarrow$ no prison, more potential partners), then small/medium size companies will adopt our approaches;

Conclusion

Hope

- DG has proved to be relevant for resource sharing ⇒ transpose this success story to the Cloud and PaaS universes ⇒ offer a technical alternate to Google, Salesforce, Amazon big farm of servers
- Our approaches are based on emerging open source Cloud solution. From an economic point of view: if it is less expensive to host services locally and if it offers more advantages (we are not “dependent on a technology” → no prison, more potential partners), then small/medium size companies will adopt our approaches;
- Main change: accept to manage redundancy, scaling the server (even for temporary needs), synchronisation ⇒ coordination with grid technology (BonjourGrid, PastryGrid?);

Conclusion

Hope

- Benefit: less expensive (comparing to Amazone EC2) because you control your data

Conclusion

Hope

- Benefit: less expensive (comparing to Amazone EC2) because you control your data
- Could also be implemented and deployed with a centralized Web Services $\Leftrightarrow$ for each application and for each Cloud type, you need a specific coordination protocol $\Rightarrow$ single point of failure.

Conclusion

Hope

- Benefit: less expensive (comparing to Amazone EC2) because you control your data
- Could also be implemented and deployed with a centralized Web Services $\Leftrightarrow$ for each application and for each Cloud type, you need a specific coordination protocol $\Rightarrow$ single point of failure.
- Ex: a company wants to install the Virtual Desktop EyeOS and the TioLive/ERP5 PaaS. During the night, the company rents different services:
  - one (company) to many – many (services) to many (companies) = new abilities, new business!
  - demonstrate that a single coordination protocol is better than configuring as many middleware than we have software!

People to thank!

At Paris XIII

Christine Choppy, Laure Petrucci , Kais Klai, Sami Evangelista, Hassna Louadah

In Tunisia

Leila Abidi, Mohamed Jemni, Heithem Abbes

People to thank!

At Paris XIII

Christine Choppy, Laure Petrucci , Kais Klai, Sami Evangelista, Hassna Louadah

In Tunisia

Leila Abidi, Mohamed Jemni, Heithem Abbes

In China

The APSCC'2010 local organizing committee

⊕ The design of BonjourGrid, a decentralized system for the coordination of multiple instances of Desktop Grid middleware

Christophe Cérin[1]
http://www.lipn.fr/~cerin/

[1]Université de Paris XIII, CNRS UMR 7030, France

APSCC'2010 invited paper