« Évolution des intergiciels & outils informatiques »

Rencontres de Branville

Vincent Garonne

Laboratoire de l'Accélérateur Linéaire

22 mai 2006

Présentation du service informatique/Groupe Devdu

Sommaire

Les tendances et futures problématiques

Résumé des tendances (voir présentation M. Jouvin)

- Vers des infrastructures massivement parallèles/distribuées
  - Ressources + petites, + puissantes, e.g Palm, multi-cœur
  - $\nearrow$ et omniprésence du réseau, e.g. WiFi Max
  - Données/informations massives et largement distribuées

Les possibles questions dans 10 ans...

1. Comment utiliser ces infrastructures pour les futures expériences de physique des particules, e.g. ILC ?
2. Quelles seront les nouveaux modèles de travail, techniques et outils dans ces environnements ?
3. But de cette présentation : Essayer d'amener des éléments de réponses, poser des questions et esquisser le rôle futur du LAL

Les tendances et futures problématiques

Résumé des tendances (voir présentation M. Jouvin)

- Vers des infrastructures massivement parallèles/distribuées
  - Ressources + petites, + puissantes, e.g Palm, multi-cœur
  - $\nearrow$ et omniprésence du réseau, e.g. WiFi Max
  - Données/informations massives et largement distribuées

Les possibles questions dans 10 ans...

1. Comment utiliser ces infrastructures pour les futures expériences de physique des particules, e.g. ILC ?
2. Quelles seront les nouveaux modèles de travail, techniques et outils dans ces environnements ?
3. Méthode : « Pour voir le futur, il faut regarder derrière soi »

Les différentes générations de grilles

Les grilles de 1^{ére} génération : DataGrid, Nordugrid et GRID3

- Approche client/serveur, basée sur le toolkit Globus 2
- Identification des principales fonctionnalités et lacunes

Les grilles de 2ème génération : EGEE, ARC et OSG

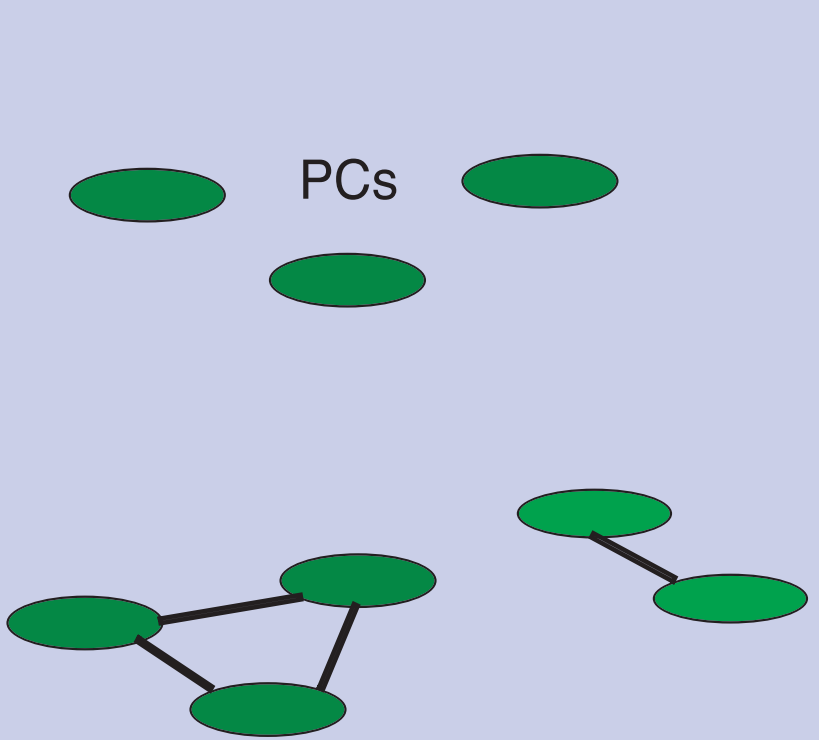
- Architecture Orientée Services, basée sur OGSI/OGSA
- Grilles de production

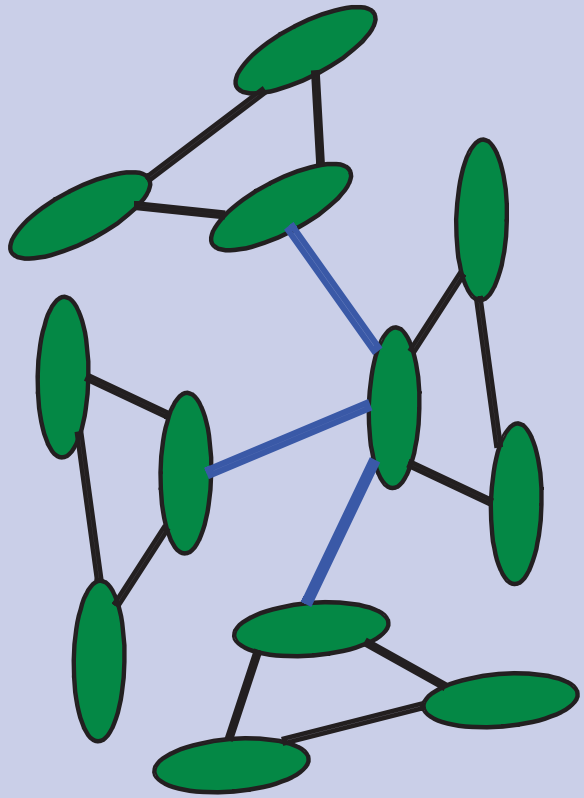
Les grilles communautaires : Alien, DIRAC et PANDA

- Développées par les expériences de physique pour leurs besoins
- Grille légère, en surcouche des grilles de production

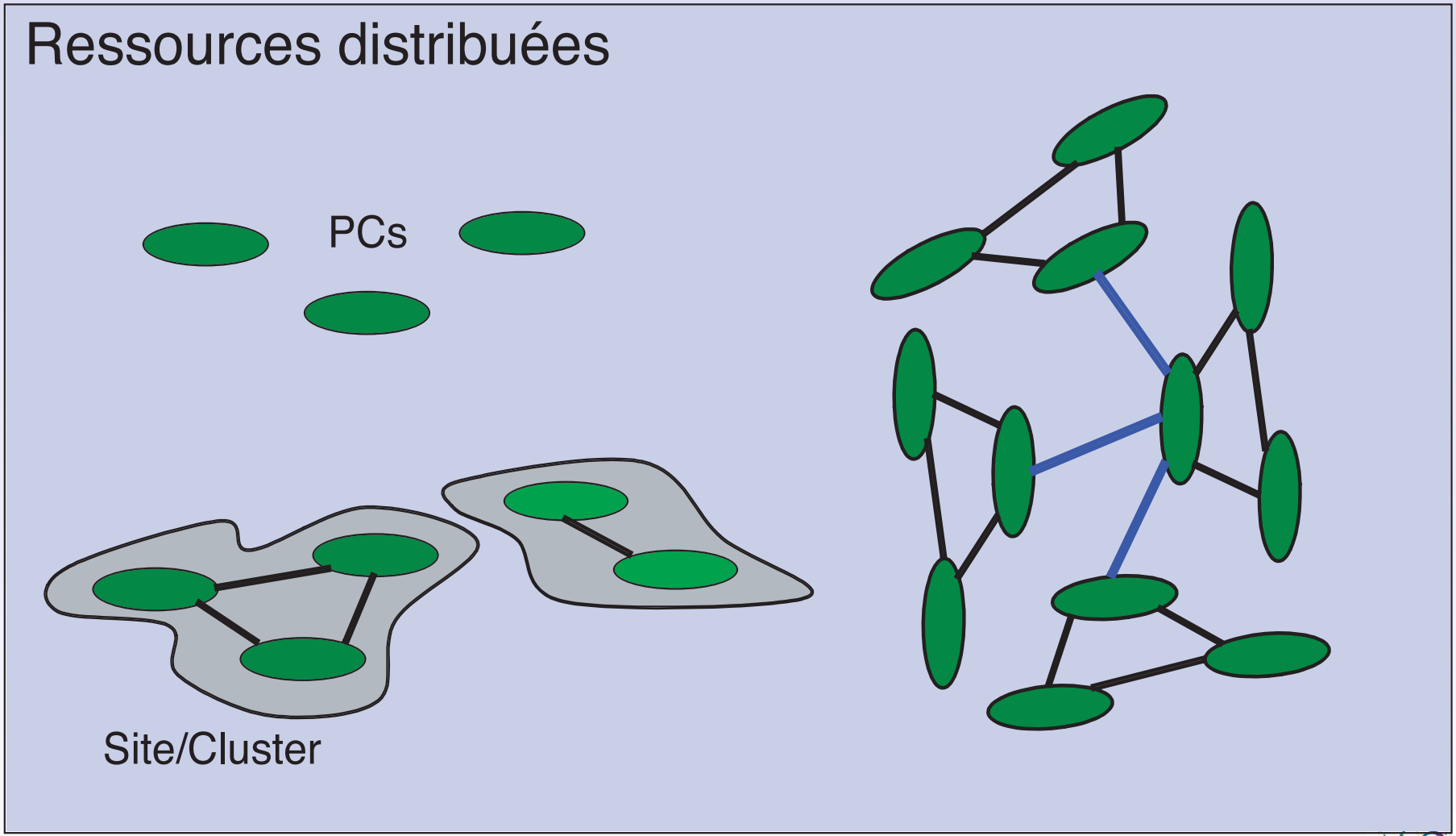
Les ressources physiques

Ressources distribuées

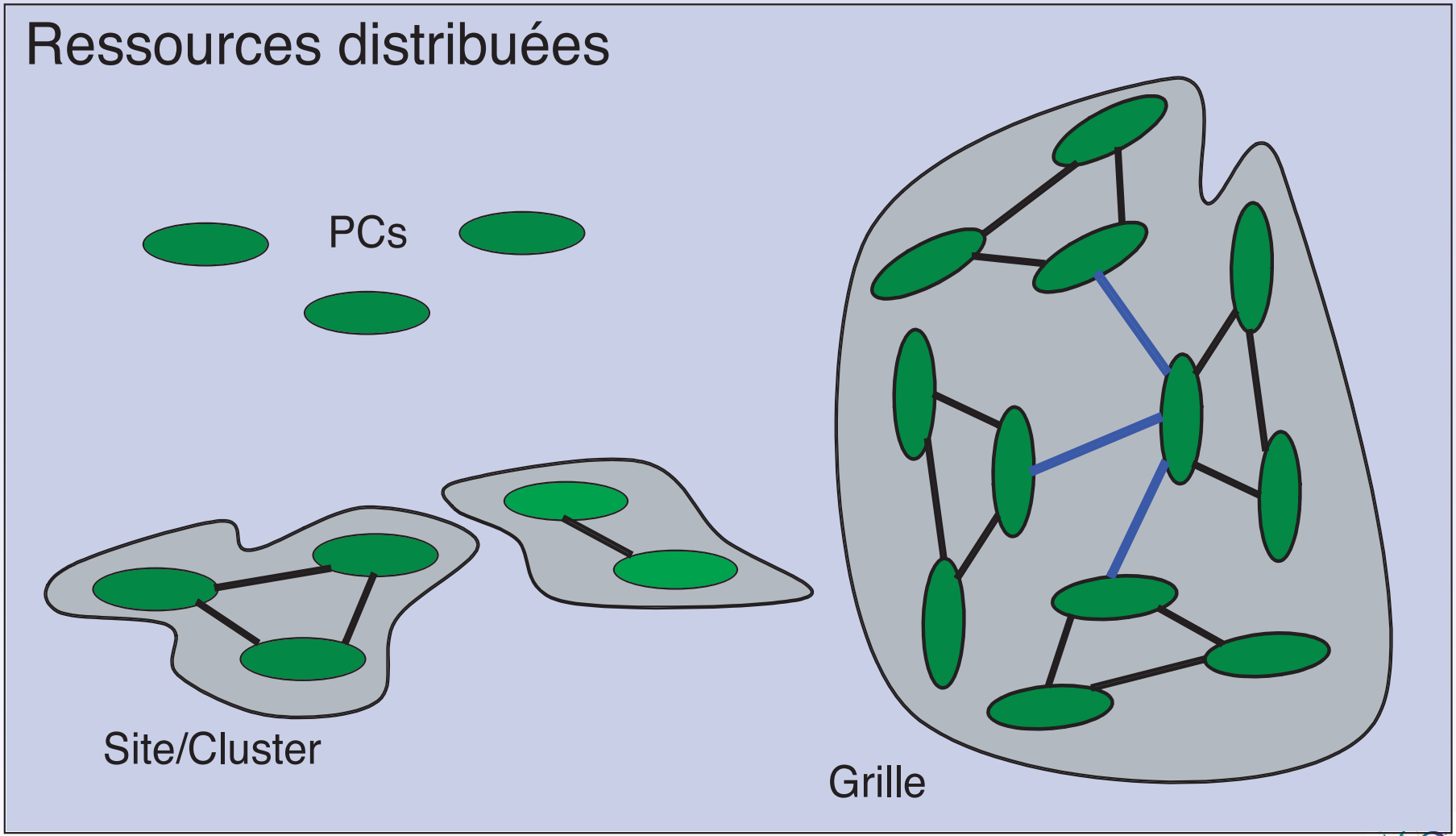




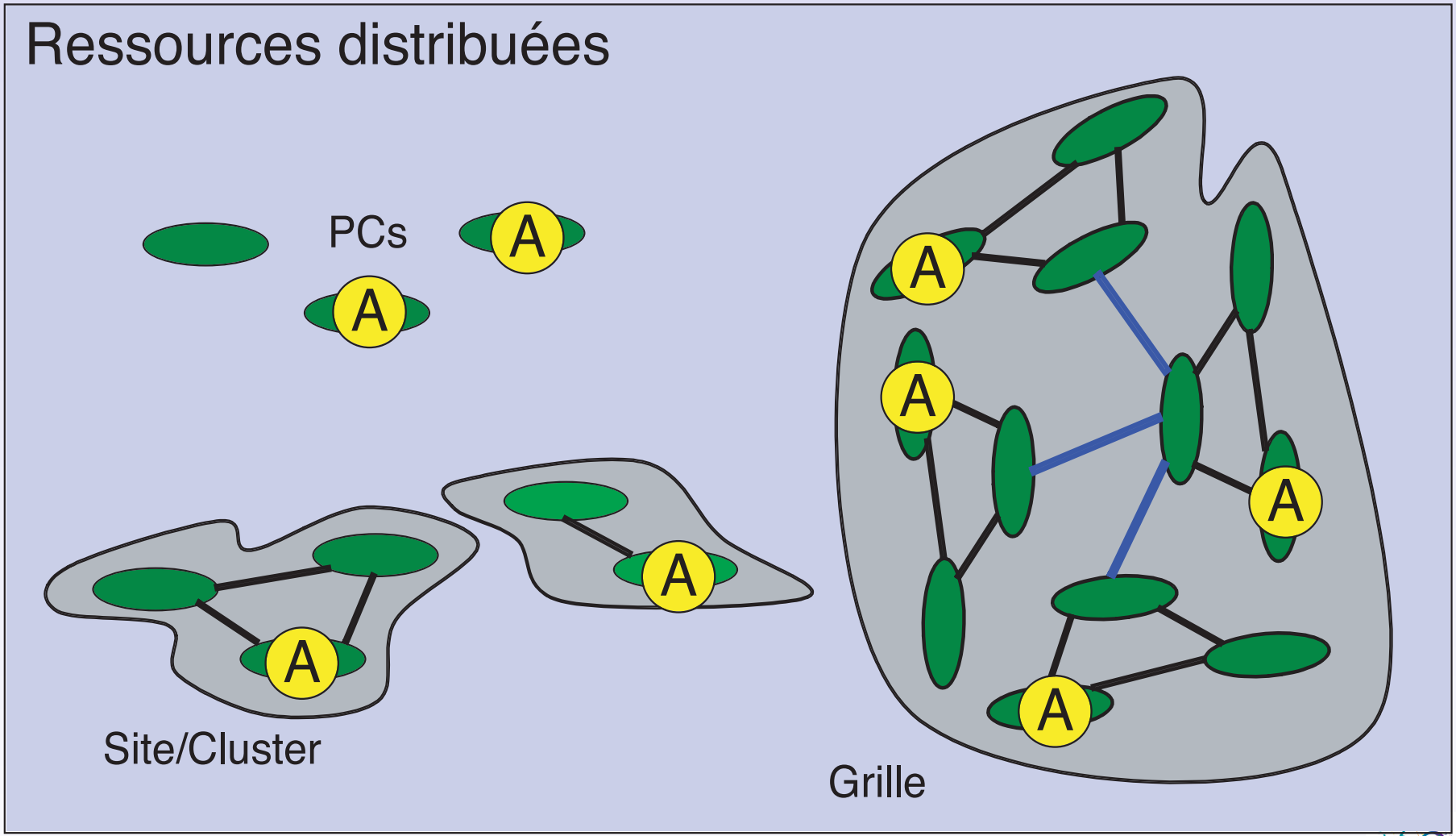
La couche logicielle locale



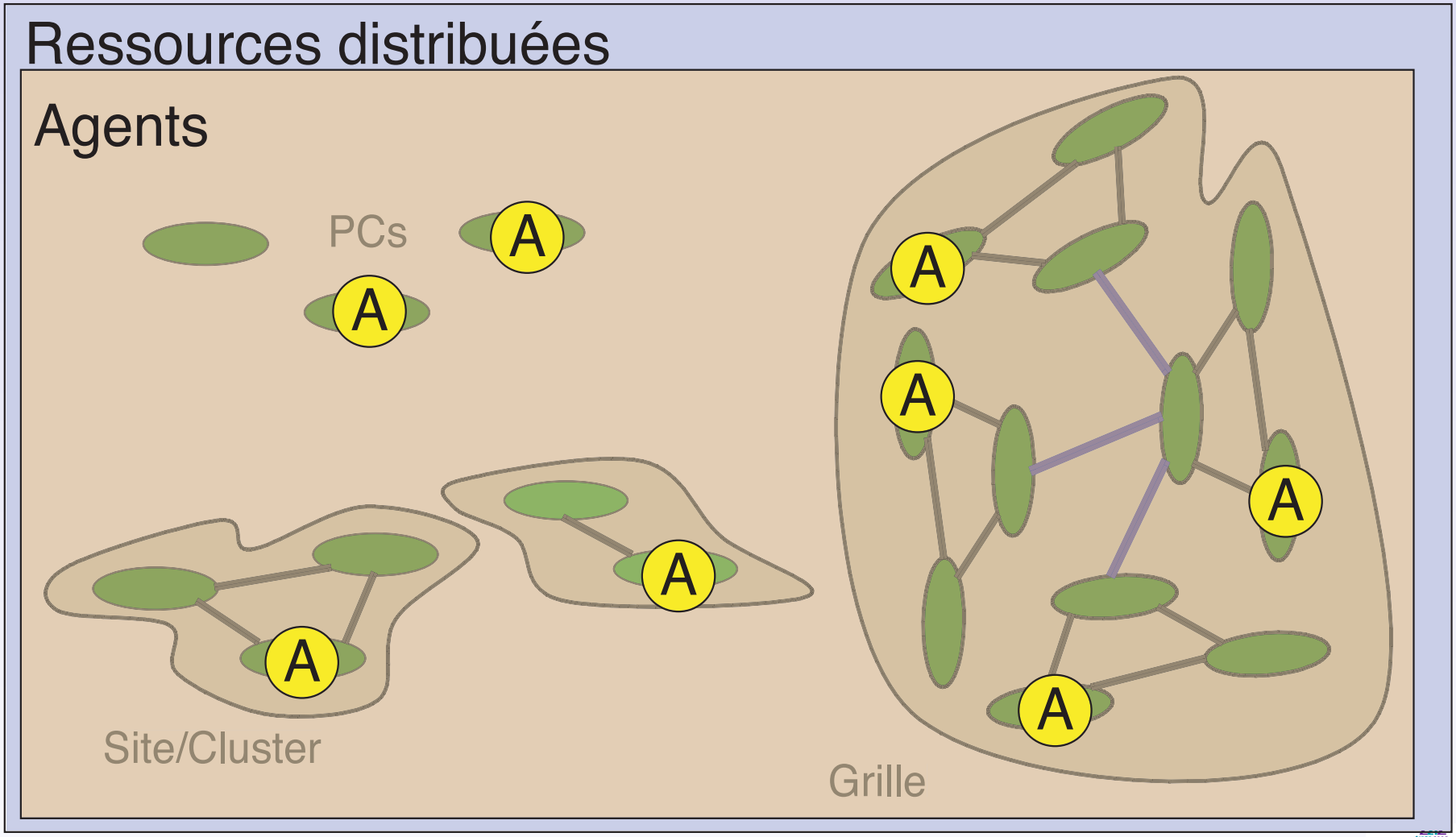
La couche logicielle globale : les grilles



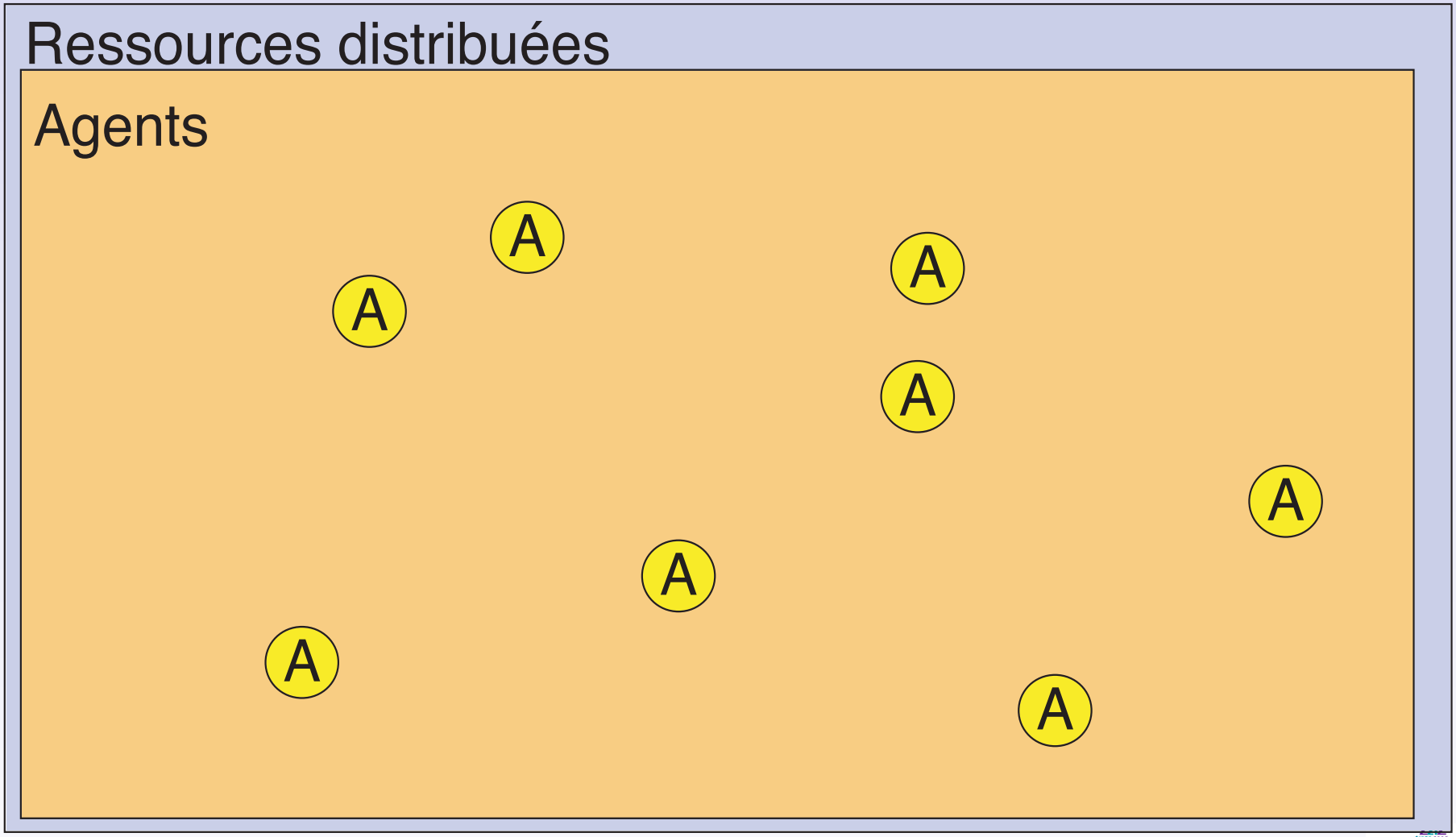
Les agents



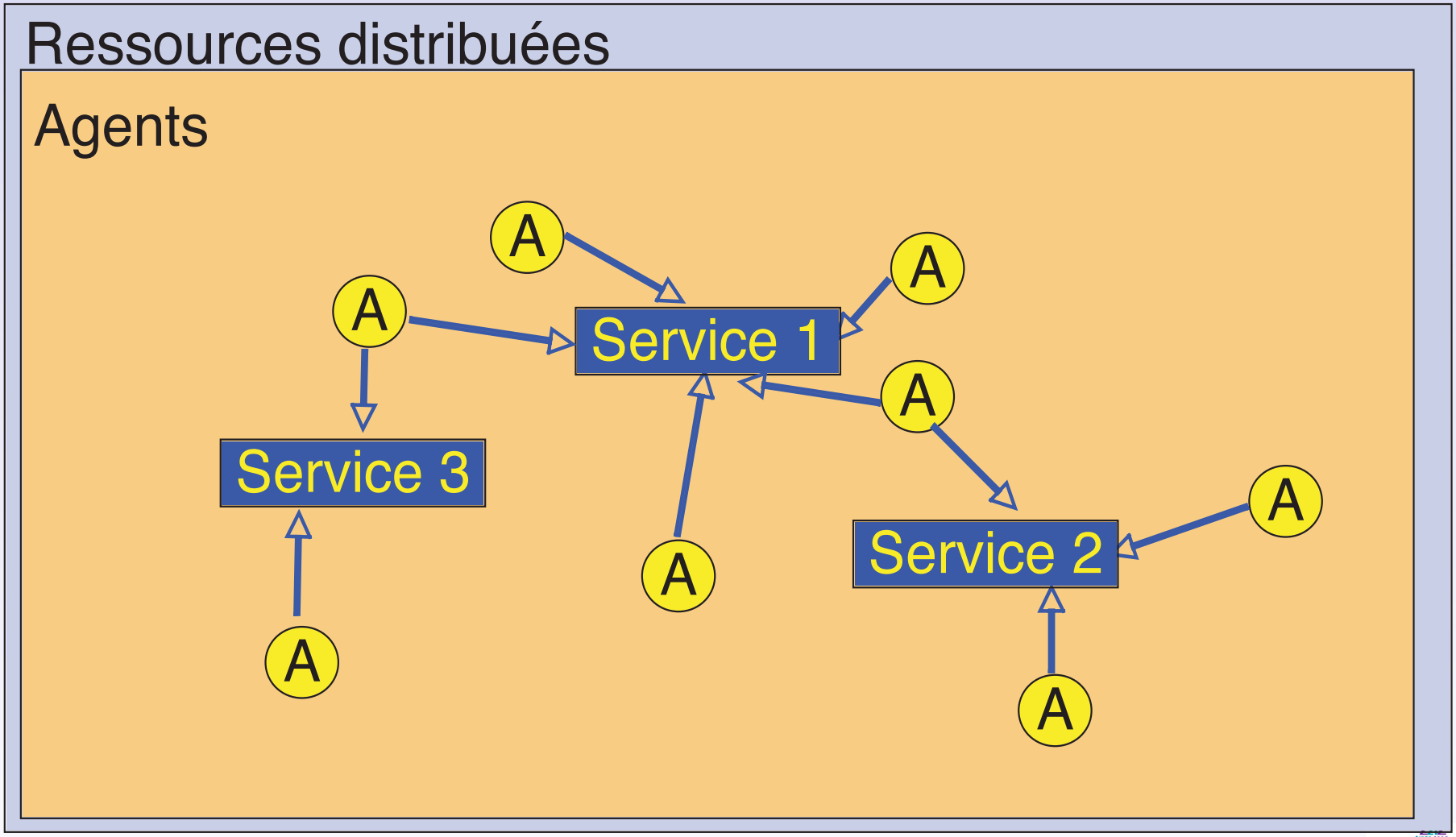
La virtualisation communautaire



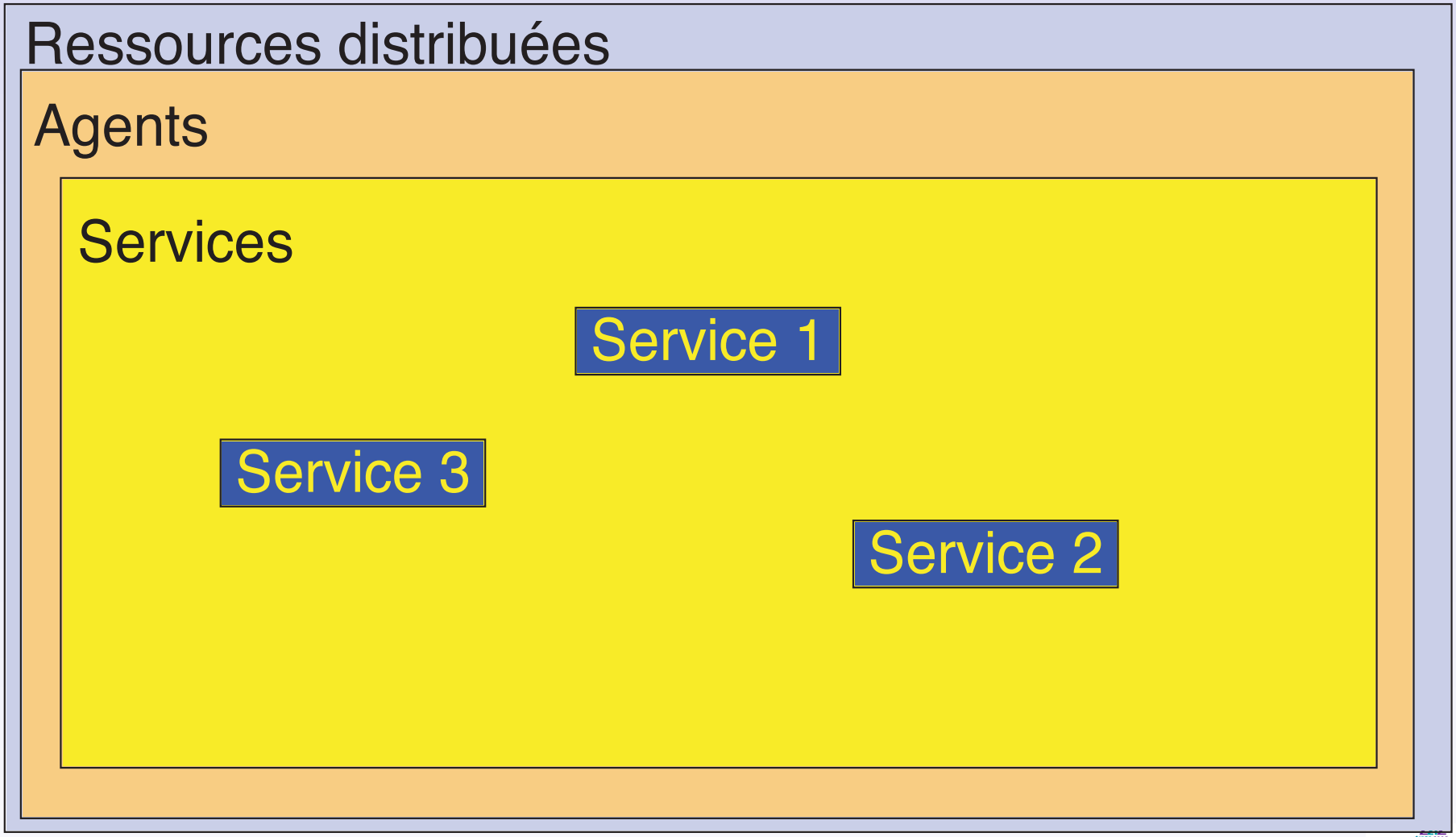
La virtualisation communautaire



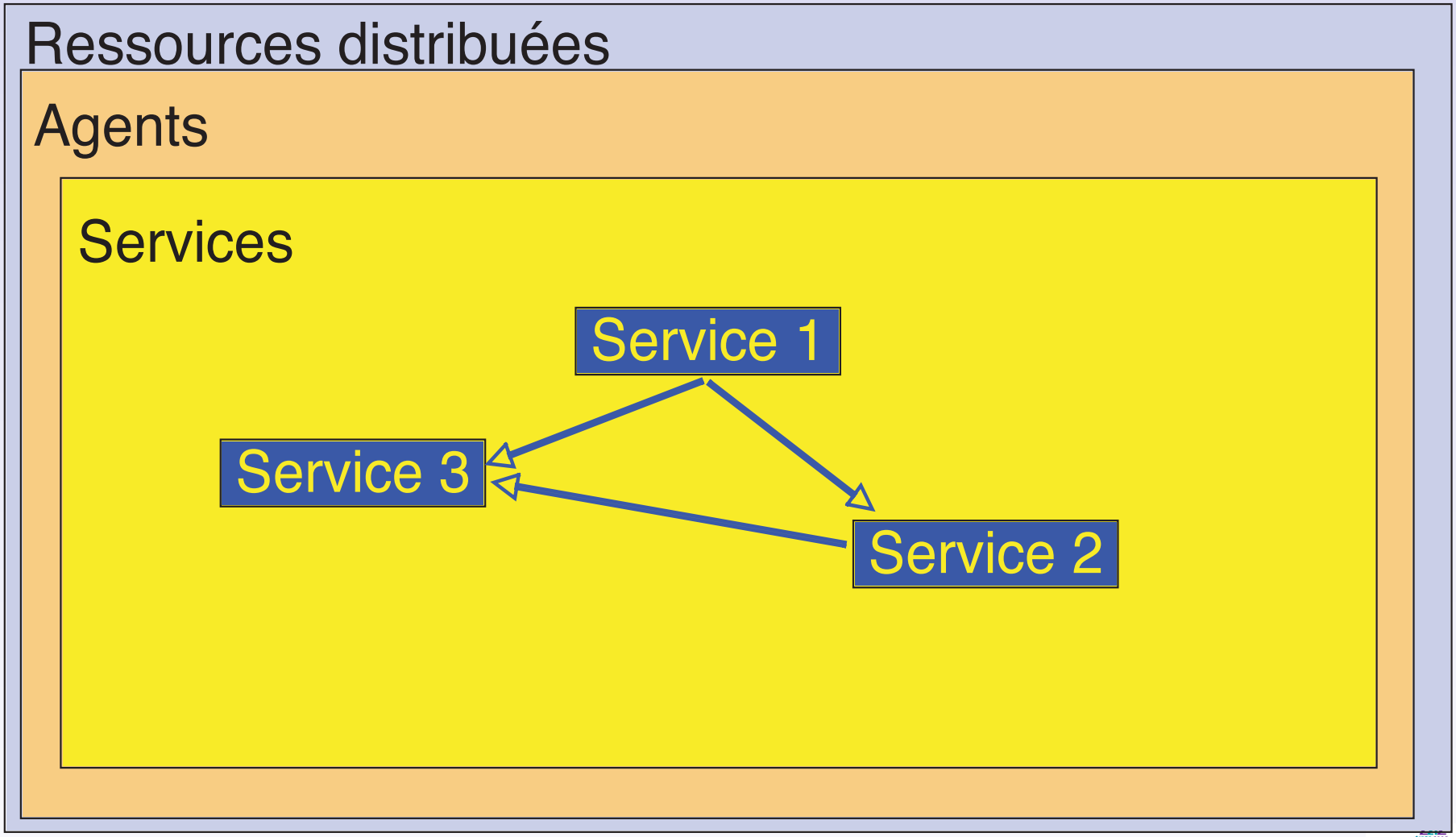
Les services



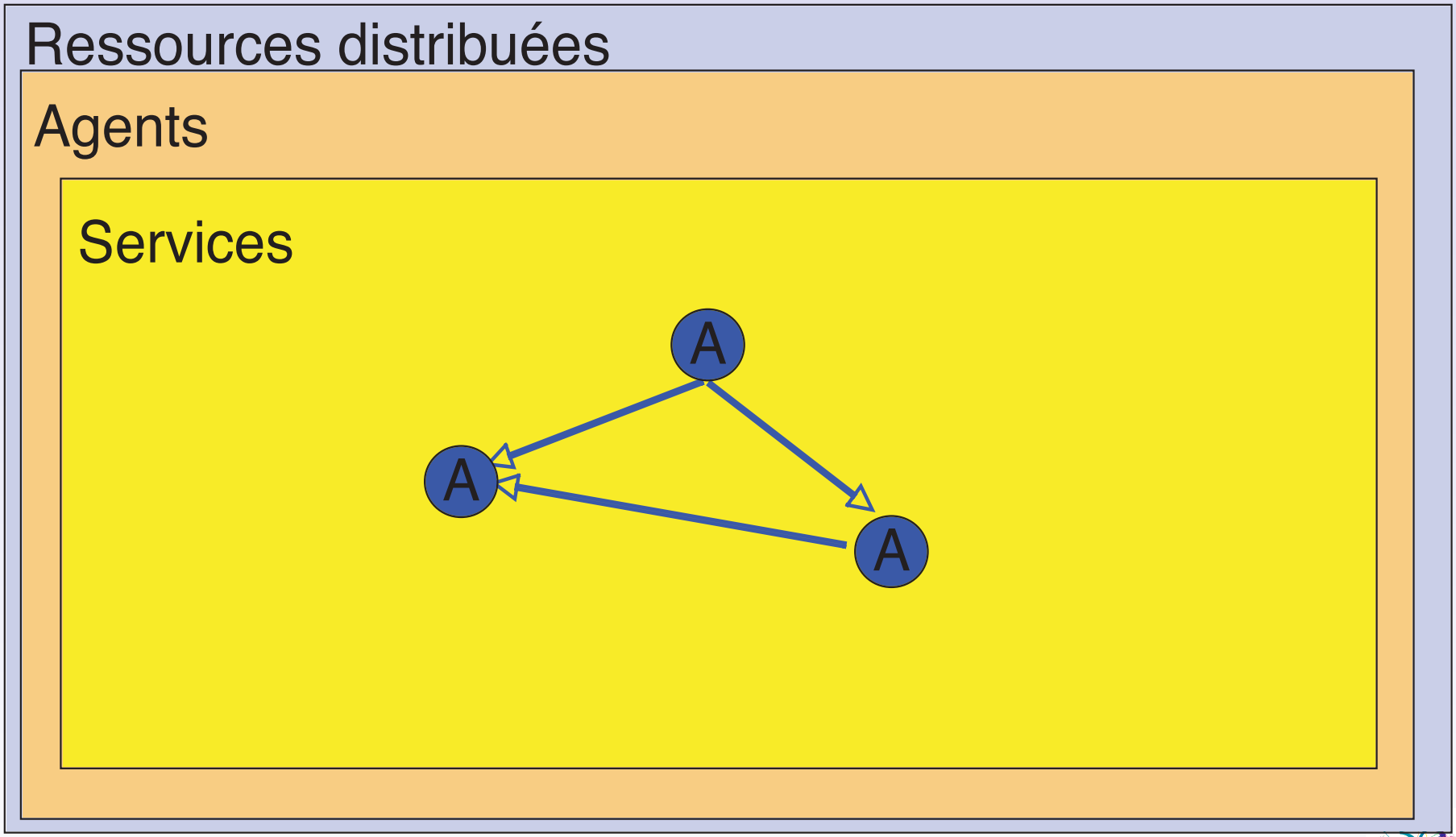
L'architecture orientée service



L'architecture orientée service



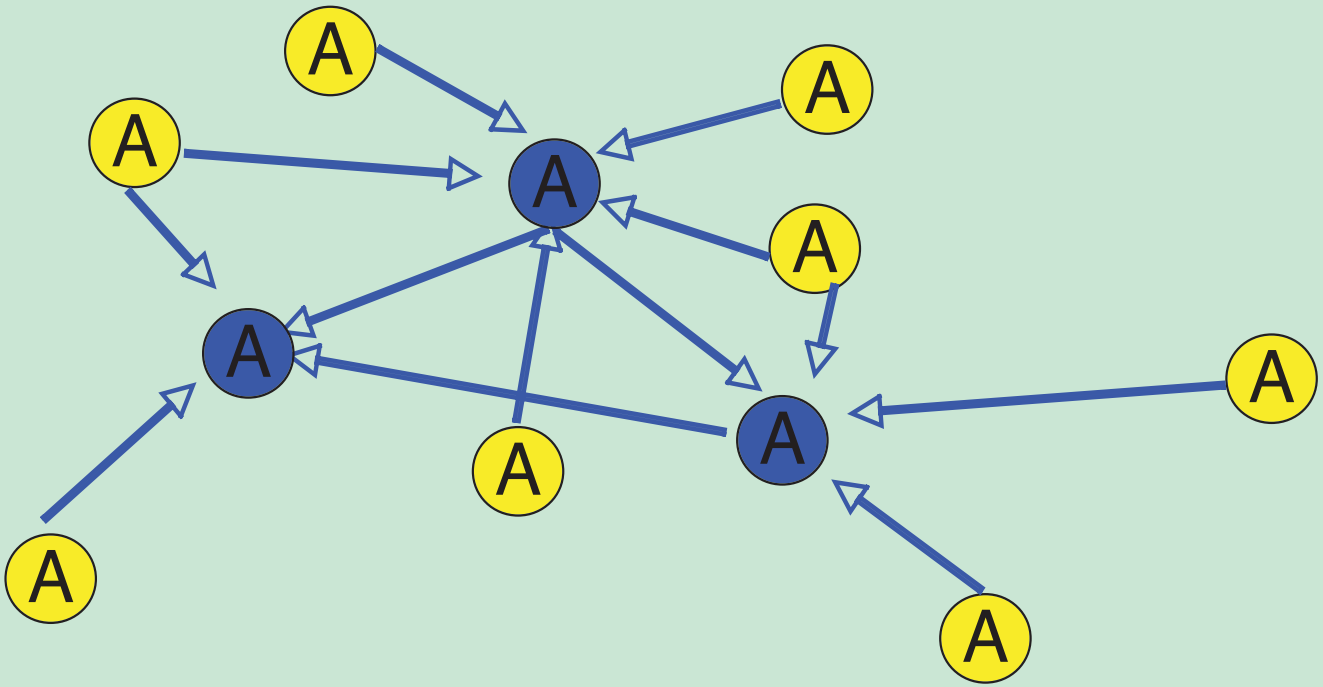
Évolution de l'architecture orientée service



Une nouvelle génération de système

Ressources distribuées

Systèmes Mutli-Agents



Les prochaines générations de grilles

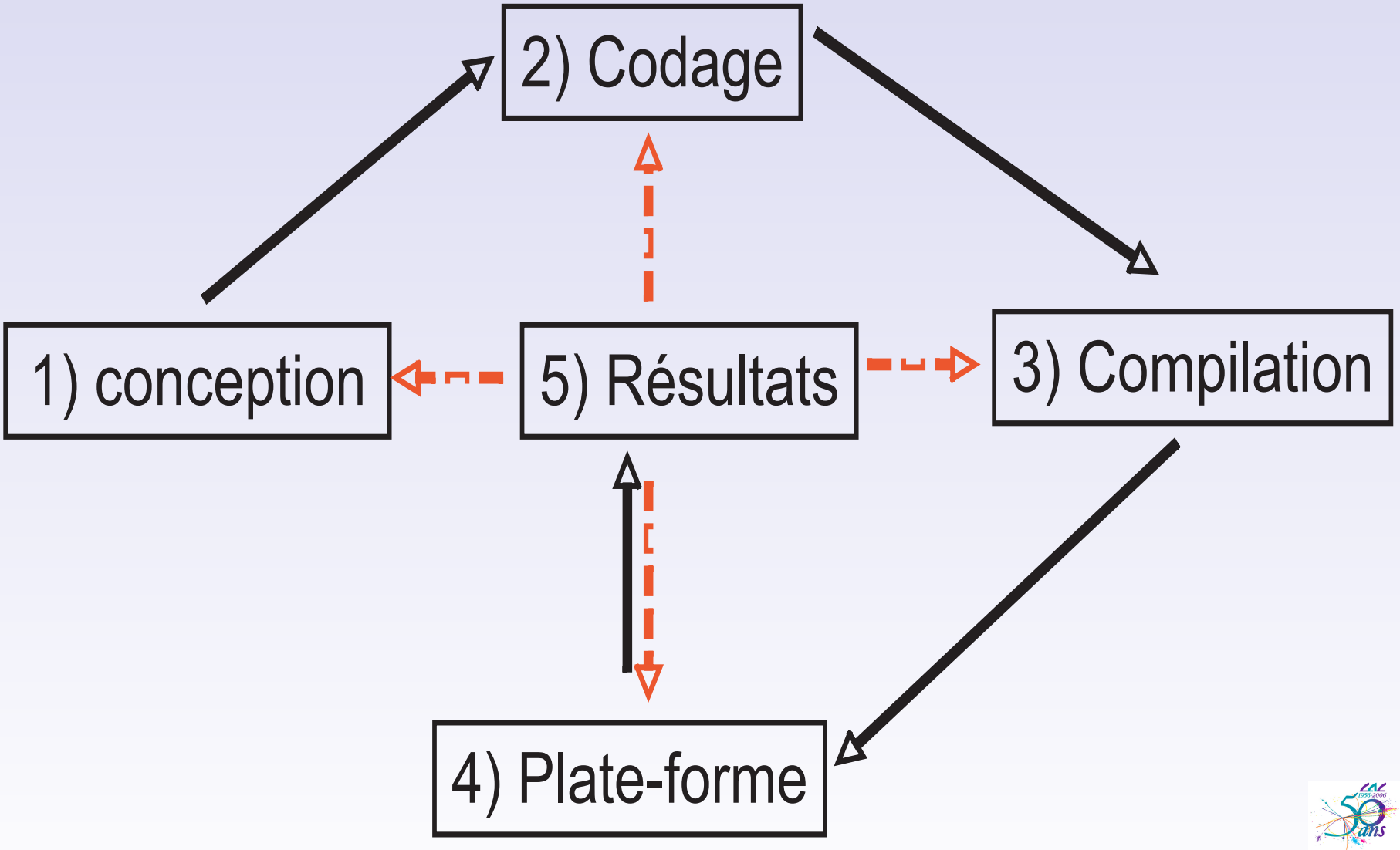
Les prédictions de Mme Irma

- Dans le court terme :
  - Convergence des grilles de production et légères
  - Plus de services identifiés, standardisés et incorporés dans l'infrastructure générique de base
- Dans le long terme :
  - le web/grille sémantique, les systèmes multi-agents et omniprésents

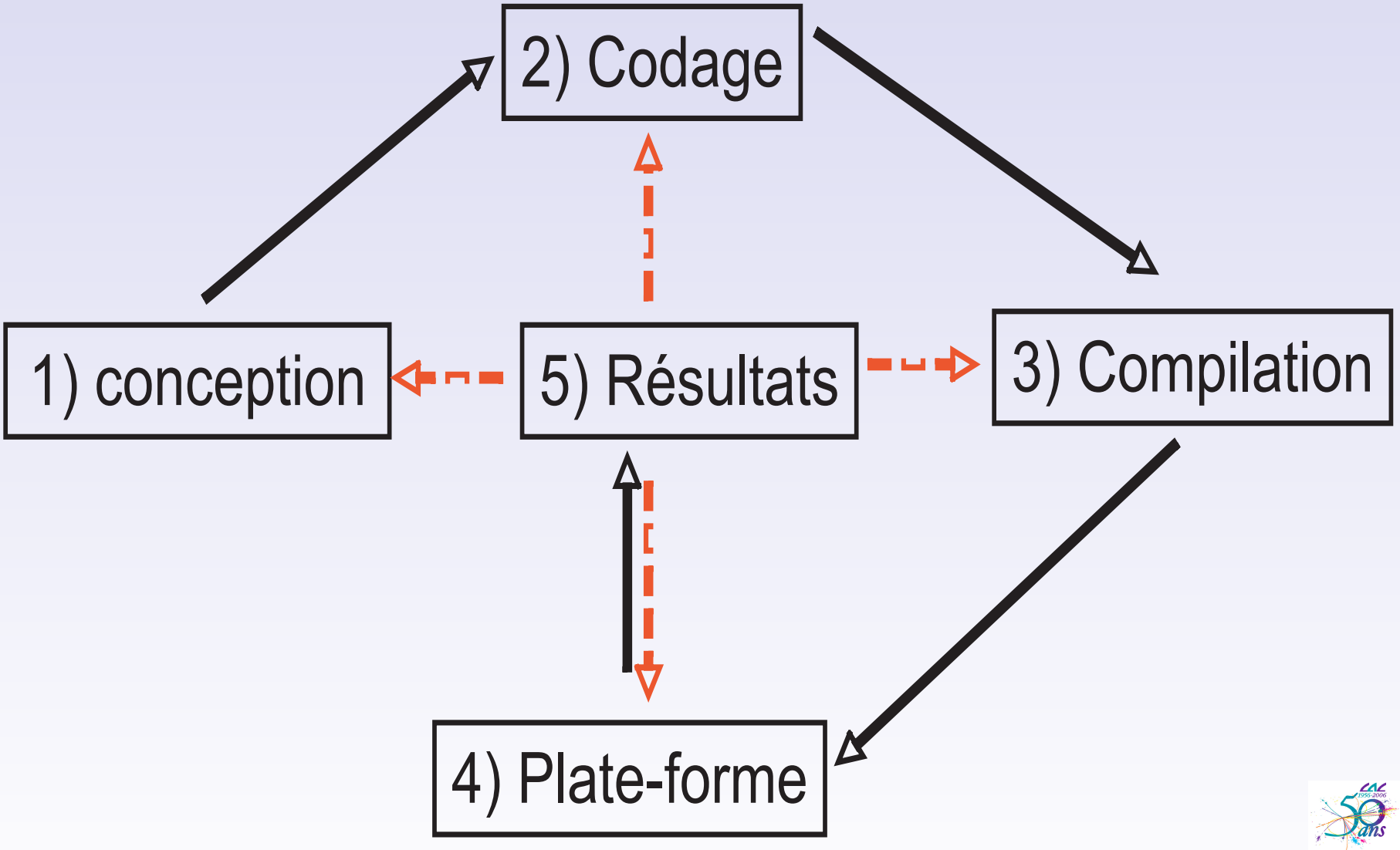
oui, très bien, mais : quel software pour 2010-2015 en HEP ?

- Comment faire de la production et de l'analyse de données ?
- Comment coupler les codes applicatifs ?
- Comment gérer les données, résultats et informations ?

Cycle de développement actuel



Adéquation dans 10 ans ?



Beaucoup de détails I

La conception

- Définition des structures de données et algorithmes
- Concepts de l'OO (UML), Découpages des classes
- Beaucoup de choses déjà faites , e.g. event model

L'implémentation et compilation

- Quel langage ? C++, C#, Java, Python ou ? ? ?
- Possibilité d'utiliser différents langages
- La réutilisabilité du code existant, e.g. Geant4
- Choix du compilateur : GNU, Intel, Pathscale, Microsoft, ...
- Mode de compilation : Archives, compilation à la volé, chargement dynamique ou statique, , ...

Beaucoup de détails II

Plate-formes & exécution : Un framework distribué et virtuel

- De plus grandes contraintes dû à un environnement distribué et hétérogène
- Développement multi-plateforme et OS
  - Pertinence de Java, Python, Ruby
  - Importance de la gestion de configuration (CMT)
- La distribution du code favorise une plus grande modularité du code
  - Quid de *Root* ? - monolithique par philosophie, *Boot* un premier pas vers plus de modularité ?
- Contraintes intrusives au niveau de l'application
  - Surveillance du déroulement de l'application
  - Traitement des erreurs applicatives

Les outils I

Évolution de l'existant

- Environnement de développement intégré
  - Visual (Windows), Xcode + Shark (MacOS X), Eclipse (Linux)
- Ajout de fonctionnalité pour le développement multi-threadé, parallèle et nouveaux modèles de programmation
- Parallélisation automatique (Programmation « Cell »)
- Comment débugger, profiler et optimiser le code d'applications parallèles
  - Problématique de la recherche en parallélisme
  - Évolution gdb, valgrind, oprofile, ...
  - Nouveaux outils d'émulation ou système grandeur nature dédié (Grid5000 ?)
- Les systèmes de version : fin de vie de Cvs pour Svn, Darcs, ...

Les outils II

Traitements des résultats et informations

- Beaucoup de ressources, conséquences : beaucoup de résultats et d'informations
- Comment les analyser, détecter les erreurs ?
  - Problématique du « result mass checking »
- Développement d'outils d'aide à l'analyse des résultats et traitements de l'information
  - Trouver des informations pertinentes dans des masses de données considérables, hétérogènes et distribuées
  - Problématique d'apprentissage, de représentation des connaissances, d'intégration de données hétérogènes, de fouilles de données

Conclusions

- Les futurs environnements apporteront dès le départ des contraintes sur la conception/implémentation du soft
- Beaucoup de problèmes se résoudront de manière pragmatique et expérimentale
- Certains concernent la recherche en informatique
- Importance du rôle des experts
- Nos convictions pour un meilleur futur :
  - Manque de comités (transversaux) d'experts de standardisation dans plusieurs domaines (services grilles, persistances des données, visualisation, etc.)
  - Besoin de standards « open source », modulaires et inter-opérables
  - « Une fonctionnalité, plusieurs implémentions » vs. ré-invention multiple de la roue

Questions ?

- Quel rôle pour le LAL ?