Déduplication extrême avec SIDUS : un premier pas vers la reproductibilité ?

Emmanuel Quemener, Loïs Taulelle

emmanuel.quemener@ens-lyon.fr, Centre Blaise Pascal, École Normale Supérieure de Lyon

lois.taulelle@ens-lyon.fr, Pôle Scientifique de Modélisation Numérique, École Normale Supérieure de Lyon

Overview

Where we are : each process or simulation launched on a computer are numerical experiments. During decades, since the mass extinction of analogical processors, we think that we live in a digital world of determinism. In fact, some introspection on floating point operations crack this point of view : numerical errors exist and become more and more unpredictable ! A new kind of variability occurs these past years : processors get more and more cores, Operating Systems launch more and more processes to keep the whole system running, all networks use non deterministic access to media, processors hold 3 layers of cache memories, dozen of ALU (arithmetic and logic unit) and control units, frequencies of processors and memories change all the time, recent processors have got only one constraint : respect a maximum Thermal Design Power ! To complete this panel, applications run on several machines (from dozens to thousands nodes) where it's very difficult to make sure that the OS, the basement of our codes, are identical from one to the other.

So, how can we reduce as much as possible the variability of our complete system, the variability in results themselves and, more oftenly, on jobs durations : network, hardware, OS, scientific background software and finally our code ? In another way, how can we reduce the port from our workstation or tiny clusters to huge ones ?

A first step could be to choose the distribution providing the most large panel of scientific stuff in order to avoid recompilation on back-office of codes : Debian (and as complete as Debian derivative distributions) is one interesting solution. The Debian policy is also one of the most drastic of the world by its quality assurance : it would be a pity not to use it !

A second step could be to provide an environment which is customized one time and deployed to many machines as fast as possible, where each “running” modification is instantaneously propagated to all ones : SIDUS for Single Instance Distributing Universal System, an operational project of Blaise Pascal Center, is one solution.

The two mains characteristics of SIDUS are : uniqueness of configuration (system provided on all “clients” is unique) and local use of resources (processors, RAM or peripherals used are the host ones : only the storage is remotely accessed).

With SIDUS on Debian Universal System, when lack of reproducibility occurs in “time” (on the same host) or in “space” (on many seems to be identical hosts), the origin must be outside the OS & scientific stuff !

First, we focus on the universal use of SIDUS in very different contexts on Scientific Computing. From a workstation available in seconds (with Compute On My Own Device approach) to large cluster nodes. Second, we describe briefly both installation and administration processes of SIDUS environment. Third, we illustrate examples of lack of reproducibility on old and recent hardwares and develop why SIDUS is a interesting solution to improve consistency of results on computation with clusters.

Déterminisme numérique : mythe ou réalité ?

Où en sommes-nous ? (ou quelle confiance pour nos instruments ?)

Chaque traitement ou simulation lancés sur un ordinateur sont des « expériences numériques ». Depuis plus de 30 ans, depuis l'extinction massive des calculateurs analogiques, nous pensons vivre dans le fabuleux monde du déterminisme numérique. En fait, une rapide introspection sur le traitement des opérations en virgule flottante altère ce point de vue : des erreurs d'arrondi affectent tous les calculs et les différentes approches de parallélisme les rendent de moins en moins prédictibles.

A cela s'ajoute désormais de nouveaux types de variabilité, corollaire d'une complexité croissante : les processeurs disposent de plus en plus de cœurs, les systèmes d'exploitation nécessitent de plus en plus de processus pour conserver actifs la totalité de leurs services courants, tous les réseaux utilisent des

méthodes d'accès non déterministes à leurs médias de communication, les processeurs embarquent trois niveaux de cache, des douzaines d'ALU (unités arithmétiques et logiques) ou d'UC (unités de contrôle), les fréquences des processeurs voire de la mémoire changent continuellement et, pour couronner le tout, les processeurs récents n'ont plus qu'une seule contrainte : respecter leur enveloppe thermique maximale (la TDP ou Thermal Design Power) ! Cerise sur le gâteau, les applications tournent sur des machines différentes (de la douzaine à des milliers) sur lesquelles il est très difficile de s'assurer qu'elles partagent exactement le même OS, au bit près.

Où voulons-nous aller ? (ou que pouvons-nous espérer ?)

Ce panorama établi, comment pouvons-nous réduire autant que possible la variabilité de notre système dans son ensemble, la variabilité dans les résultats eux-mêmes, et, plus fréquemment, la variabilité sur les durées d'exécution : le réseau, les matériels, l'OS, le socle logiciel et finalement notre code ? Dans le même ordre d'idée, comment pouvons-nous réduire l'effort de portage de notre station de travail ou de petits clusters départementaux vers les grosses machines (méso, nationales, européennes) ?

Comment y allons-nous ? (ou la meilleure solution que nous ayons trouvée)

La première étape est de choisir « la » distribution offrant le plus large panorama possible de logiciels scientifiques précompilés et préconfigurés pour éviter à tout prix d'inutiles recompilations et réinstallations du socle de nos codes : la distribution Debian (ou toute autre distribution aussi complète que la Debian) est une approche pragmatique. L'assurance qualité de Debian est aussi une des plus tatillonnes qui soit : ce serait dommage de ne pas l'utiliser et de réinventer la roue.

La seconde étape serait de fournir un environnement configuré une seule et unique fois et déployé sur tous ses clients, où chaque modification est instantanément propagée sur toutes les autres : SIDUS pour *Single Instance Distributing Universal System*, un projet pleinement opérationnel au Centre Blaise Pascal (maison de la simulation lyonnaise) sur une centaine de machines permanentes et au Pôle Scientifique de Modélisation Numérique (un des méso-centres lyonnais) sur plus de 330 nœuds.

En résumé, les deux principales caractéristiques de SIDUS sont :

- l'unicité de sa configuration : l'OS proposé à tous les clients est unique ;
- l'utilisation des ressources locales : processeurs, mémoires et périphériques utilisés sont ceux des clients. Seul le stockage est accédé de manière distante.

Retour à la reproductibilité : « si c'est pas moi c'est lui... »

En conséquence, avec SIDUS associé au système d'exploitation universel Debian, quand des défauts de reproductibilité apparaissent, « dans le temps » (sur un même nœud) comme « dans l'espace » (sur des nœud différents), la source ne peut pas, par essence, en être attribuée au système puisque c'est le même au bit près ! Ainsi, l'origine de cette variabilité (et nous en avons trouvé bien malgré nous) est à rechercher hors de l'OS ou du socle de calcul scientifique.

Qu'allons nous présenter ?

En premier lieu, nous détaillons l'usage de SIDUS dans les différents contextes de calcul scientifique que nous avons explorés, d'une station de travail disponible en quelques secondes avec COMOD (pour *Compute On My Own Device*) à des clusters de plusieurs centaines de nœuds. Puis, nous abordons brièvement l'installation d'une instance de SIDUS et quelques tâches d'administration typiques. Ensuite, nous illustrons les défauts de reproductibilité que nous avons constatés autant sur les matériels récents que sur les anciens. Enfin, nous concluons pourquoi SIDUS est un outil résolument intéressant autant pour se simplifier son travail d'administrateur de plate-forme que pour assurer, auprès de ses utilisateurs, la pertinence de ses résultats.

Enjeux scientifiques, besoin en calcul, stockage et visualisation

Initialement, le principal enjeu de SIDUS est technique, voire pragmatique : en limitant toutes les phases d'administration du système d'exploitation ou de son déploiement, en la réduisant aux opérations d'une sommaire station de travail, SIDUS permet à l'administrateur de se focaliser sur d'autres tâches, plus en relation avec le calcul scientifique ou l'assistance aux utilisateurs.

De ce fait, plus aucune opération d'installation de système ou d'administration groupée des équipements n'est nécessaire : toutes les machines exécutent le même système et toute modification sur le socle est instantanément répercutée à tous les clients qui l'exploitent. Balayée l'administration lourde de support de

stockage : en effet, le retour d'expérience sur ces 5 dernières années montre que la majorité des opérations de maintenance matérielle concerne les disques durs. De plus, en supprimant les disques durs des nœuds, ce sont de précieux Watts (de 3W et 20W par disque) que nous économisons et des BTU (*British Thermal Unit*, unité classique exprimant une dissipation thermique) que nous récupérons.

D'autre part, face aux évolutions matérielles permanentes nous imposant toujours plus de vigilance, un outil permettant la comparaison rapide entre deux machines différentes, de paramètres de BIOS, de noyaux voire d'environnements climatiques différents permet de juger instantanément des différences et donc statuer de la solution à choisir entre différents scénarii, cela en s'affranchissant totalement d'une installation atomique !

Également, dans l'apprentissage du calcul scientifique, SIDUS propose de faciliter l'accès aux ressources. Si les stations de travail individuelles sont rares et difficilement configurables, SIDUS se déploie avec la même instance que ce soit sur un poste physique où à l'intérieur d'une machine virtuelle : l'utilisateur conserve son propre OS et démarre son client SIDUS en quelques secondes, en allouant les ressources qu'il souhaite (cœurs et mémoire vive). En outre, un dossier d'échange assure le partage de données entre hôte et client SIDUS.

Ainsi, le laboratoire de chimie de l'ENS-Lyon depuis plus d'un an et les trois dernières sessions de *Computational Physics* de l'école physique des Houches (2011, 2012, 2013) ont exploité SIDUS dans cette configuration de machines virtuelles : plus de temps perdu à installer un environnement sur chaque poste personnel ; un démarrage de l'instance SIDUS suffit. Pour cette approche « COMOD », les principales utilisations ne se cantonnaient, curieusement, pas seulement aux simulations courtes : au delà du prototypage vers de plus grosses structures de calcul ou la réduction de données, ce sont des simulations de plusieurs semaines qui ont offert à SIDUS un gage de stabilité, même dans un environnement « ouvert ». Grâce à l'universalité de SIDUS, la transition vers des ressources plus étoffées est instantanée : les stations de travail et les nœuds du Centre Blaise Pascal partagent le même système et le centre de calcul de l'ENS-Lyon, le PSMN, utilise maintenant une instance de SIDUS sur plus de 330 nœuds en production (dont le dernier Equip@Meso).

Développements, utilisation des infrastructures

Avant d'entrer dans le cœur du fonctionnement de SIDUS, il convient de préciser ce que SIDUS n'est pas :

- SIDUS n'est pas *Linux Terminal Server Project :* LTSP propose un client léger démarrant un serveur X. Les ressources exploitées (à l'exception de l'affichage graphique) restent celles du serveur ;
- SIDUS n'est pas *Fully Automatic Installation* ou Kickstart : FAI ou Kickstart fournissent un ensemble d'outils permettant l'installation complète d'un système sur le disque local . Chaque machine est installée rapidement mais d'autres outils de synchronisation sont nécessaires comme cfengine, puppet ou chef ;
- SIDUS n'est pas un LiveCD démarrant sur réseau : un LiveCD permet le démarrage d'un environnement complet, mais il est figé et difficile à mettre à jour ou à propager instantanément.

SIDUS n'utilise que des logiciels *Open Source* pérennes associés à des protocoles standardisés : DHCP, PXE, TFTP, NFS, NFSroot, AUFS en sont les mots clés. C'est cette dernière « astuce » de LiveCD, AUFS, qu'exploite SIDUS pour offrir le même système fonctionnel à tous ses clients : AUFS, pour *Another Union File System*, permet la superposition d'une arborescence en lecture seule fournie par NFS (notre OS), et un système de fichiers en lecture/écriture placé en mémoire vive. Tout se passe comme si, au final, le système était « classique » : toute modification sur le système est possible. Les modifications disparaissent évidement au redémarrage. Si la persistance entre deux redémarrages est indispensable, SIDUS offre également plusieurs solutions de « persistance », basées soit sur NFS, soit sur iSCSI (*Internet Small Computer System Interface*, protocole de partage de volume de stockage sur IP). A noter qu'il convient d'être vigilant pour maintenir une cohérence entre l'instance unique et chaque modification atomique.

Parmi toutes les solutions permettant le démarrage en réseau pour l'offre de système de stockage, SIDUS est certainement celle exigeant le moins de ressources. Le premier serveur SIDUS disposait de 40 Go d'espace disque, d'une connexion Gigabit Ethernet, de 4 Go de mémoire et 2 processeurs très anciens. Grâce à l'utilisation de NFS et son « mode fichier », les éléments à « livrer » sur le réseau pour un nouveau client sont identiques au précédent client : le « *boot storm* » tant redouté se limite alors à la capacité du serveur à service son cache mémoire : il n'en est pas de même pour les déploiements de disques réseau en « mode bloc ».

Ainsi, l'usage de SIDUS dans les infrastructures de calcul s'appuie sur un retour d'expérience de plus de trois années, lequel a vu la croissance du parc de quelques nœuds ou postes de travail à des centaines de nœuds d'équipements parfois hétérogènes. L'intérêt de SIDUS réside également dans sa capacité à

s'adapter à un environnement quelconque : il suffit que les commandes IPMI (*Intelligent Platform Management Interface*) permettent le redémarrage sur le réseau, qu'une instance SIDUS soit disponible et tous les nœuds pourront démarrer une instance SIDUS. Dans le cadre de campagne de tests, il est alors possible, rapidement, sans temps d'intégration, de lancer ses propres bancs d'essais sur ses systèmes : nous disposons, ainsi, véritablement, d'une comparaison pertinente entre différentes infrastructures. Plus de temps de perdu à configurer les BIOS et à installer les systèmes sur des supports de stockage capricieux : tout se réalise par le réseau.

Outils, complémentarité des ressources, difficultés rencontrées

Par essence, le Centre Blaise Pascal est une maison de la simulation disposant de nombreux plateaux techniques : multi-nœuds, multi-cœurs, GPU, architectures exotiques, distributions Linux passées, présentes et futures. L'essentiel de ces ressources matérielles brillait par son hétérogénéité mais la robustesse de SIDUS et sa capacité à démarrer en réseau tous ses équipements en font, tacitement, le socle d'une majorité de ses ressources : d'un usage épisodique de quelques minutes comme station de travail sur un poste déjà fonctionnel à celui de stations de travail dédiées ou de nœuds de calcul hétérogènes (jusqu'à 6 architectures de processeurs différentes), la seule différence que distingue l'utilisateur réside dans l'usage de l'outil de soumission. La polyvalence de SIDUS a aussi été largement éprouvée lors de l'intégration de composants matériels spécifiques comme les interconnexions InfiniBand et les cartes GPGPU (*General Purpose Graphic Processing Unit*, processeur graphique utilisé à des fins généralistes). Sans souci particulier, juste de petits paramétrages complémentaires pour en exploiter toute la richesse.

Basé sur une Debian depuis les origines, l'installation de SIDUS se déroule en huit étapes (dont les 1,6 et 7 sont fondamentales) :

1. formation d'une racine d'un système Debian par Debootstrap
2. Création d'un « cordon ombilical » avec le système hôte
3. Installation des paquets complémentaires (et purge de quelques uns...)
4. Adaptation à l'environnement local (langue, clavier, fuseau horaire)
5. Pointage vers les services tiers (identification/authentification, dossiers utilisateurs)
6. Création de la séquence de démarrage inirtd avec AUFS
7. Importation des noyau & initrd sur le serveur TFTP
8. Détachement du système hôte

La seule difficulté vient des applications qui, parfois, sont capricieuses à l'intégration dans un environnement *chrooté* sur le serveur de SIDUS. Ce sont ces contraintes qui exigent souvent l'intrication avec le système hôte (partage du même /proc ou /sys).

Résultats scientifiques

La contribution de SIDUS dans des résultats scientifiques se limite à l'environnement qu'elle propose : en assurant que le système démarré sur deux machines ou sur une même machine à deux moments différents est identique au bit près (si l'instance n'a pas été modifiée entre temps), SIDUS établit un socle sur lesquels les codes voire les outils d'investigation peuvent s'exprimer pleinement, de manière pertinente, afin d'évaluer les problèmes liés à la reproducibilité ou à la consistance des calculs.

A partir des expériences menées ces trois dernières années, nous pouvons affirmer que les sources de cette variabilité sur les temps de calcul sont multiples pour les équipements anciens : cela va de l'état des ventilateurs, du vieillissement des alimentations, du taux de remplissage des messages de la pile IPMI, de la tenue dans le temps de la pâte thermique entre processeurs et radiateur, de la position dans la baie, etc : ces différences influençaient les environnements électrique et thermique. De tels impacts nous ont surpris !

Plus inquiétant : à titre d'exemple, lors de l'évaluation de l'usage de GlusterFS comme espace temporaire haute performance sur InfiniBand, sur des équipements dernier cri, des expériences ont montré une forte variabilité dans les résultats de tests IOZone3 (réalisés pour l'occasion sur des RamDisk pour éviter tout effet lié à la latence physique des disques) : cette variabilité avait pour origine la disparité de réglages entre certains nœuds, aux valeurs d'usine très différentes (de « Maximum Performance » à « OS Controlled »).

Plus près de nous, il y a quelques semaines, lors de l'intégration opérationnelle de GlusterFS sur l'Equip@Meso de l'ENS-Lyon, des variabilités d'un ordre de grandeur affectaient la communication entre deux nœuds via InfiniBand en IP. Trouver le bon paramétrage de BIOS (différent de celui évoqué ci-dessus)

a été long et fastidieux mais a permis de limiter la variabilité à une valeur de quelques pourcents. Ainsi, les mécanismes d'autoconfiguration énergétiques, trop « laissés » au processeur, génèrent une variabilité dont l'écart-type laisse perplexe : les intégrer dans le processus même de l'expérience numérique devient une exigence !

Perspectives

Pour l'instant, SIDUS est pleinement opérationnel au Centre Blaise Pascal. Il sert de l'ordre d'une centaine de nœuds, d'une vingtaine de postes de travail et une dizaine de postes COMOD. Des laboratoires, notamment de biologie et de mathématiques, en plus du laboratoire de chimie, ont demandé son exploitation pour des postes de travail virtualisés.

Cependant, son principal utilisateur (en terme de volume de nœuds, de cœurs agrégés, de mémoire vive) reste le Pôle Scientifique de Modélisation Numérique, le centre de calcul de l'ENS-Lyon. Mis en œuvre par son personnel technique à partir de la seule documentation, SIDUS est pleinement opérationnel et a su s'adapter à un centre de calcul naguère cantonné aux méthodes traditionnelles (installation à base de Kickstart).

Hors des murs de l'ENS, la première opération sera la promotion de SIDUS au sein de la communauté HPC de région Rhône Alpes, de manière à étendre le spectre des personnes intéressées par le déploiement de clusters de toutes tailles ou de postes COMOD.

Le second axe de travail sera d'offrir un environnement COMOD partout, via une connexion VPN plus lente, pour que les utilisateurs de ressources puissent, hors site, utiliser pleinement l'outil.

L'autre axe visera à fournir un image SIDUS complète pour un environnement de grilles : l'utilisateur développe son application dans sa propre instance et charge cette dernière dans les infrastructures exactement comme dans le projet StratusLab. Elle se déploie ensuite sur les nœuds mis à disposition : un cluster à la demande exactement sur le même modèle que la station de travail à la demande. A la clé, la suppression complète des temps de portages.

Références

Présentation Distonet de JRES 2011 : https://2011.jres.org/archives/126/index.htm

Site de Sidus au CBP : http://www.cbp.ens-lyon.fr/sidus/

Présentation Scipy 2013 : http://conference.scipy.org/scipy2013/presentation_detail.php?id=199

Variabilité FPU : http://software.intel.com/sites/default/files/article/164389/fp-consistency-122712_1.pdf

Article Linux Journal : Linux Journal, November 2013, 235 pp 100-110

Poster SIDUS aux JRES 2013 : https://conf-ng.jres.org/2013/planning.html

Utilisateur SIDUS : Citation dans ACS Nano, 2013, 7 (6), pp 5273–5281