La visualisation de traces, support à l'analyse, déverminage et optimisation d'applications de calcul haute performance

Damien Dosimont[*,**], Guillaume Huard[*,***], Jean-Marc Vincent[*,***]

[*]prénom.nom@imag.fr
[**]INRIA
[***]Université Joseph Fourier, Grenoble I

Résumé. L'analyse du comportement d'applications logicielles est une tâche de plus en plus difficile à cause de la complexité croissante des systèmes sur lesquels elles s'exécutent. Alors que l'analyse des systèmes embarqués doit faire face à une pile logicielle complexe, celle des systèmes parallèles doit être capable de s'adapter à l'envergure de leur architecture matérielle et à leur indéterminisme. La visualisation de traces obtenues lors du déroulement des applications s'exécutant sur ces plate-formes s'est répandue dans les outils d'analyse pour faire face à ces problématiques. Il existe aujourd'hui un large éventail de techniques qui se distinguent par la quantité d'informations, l'échelle des systèmes, ou les comportements qu'elles sont capables de représenter. Nous nous proposons d'en faire un état de l'art, en discutant des méthodes de visualisation statistiques, comportementales et structurelles de l'application, et des techniques permettant le passage à l'échelle de l'analyse.

1 Introduction

L'analyse du comportement d'applications logicielles à des fins de compréhension, de déverminage et d'optimisation est une tâche de plus en plus difficile, en raison de la complexité croissante des systèmes sur lesquels elles sont exécutées. Dans le cas des systèmes embarqués et des ordinateurs personnels à processeurs multi-coeurs, évoqué par Herlihy et Shavit (2008), les principales difficultés sont liées à la pile logicielle complexe, car l'analyse concerne aussi bien les aspects bas-niveau proches du matériel, que les couches intergicielles abstrayant celui-ci (Campbell et al., 1999), ou que l'applicatif. Les systèmes parallèles, et en particulier les systèmes dédiés au calcul, se voient quant à eux distribués dans le monde, à travers une hiérarchie complexe (processeur, machine, grappe, site) et de plus en plus souvent hétérogène (Hwang et al., 2012). L'échelle de ces systèmes est contraignante pour les outils d'analyse. Pour tirer au mieux parti des multiples unités de calcul, les applications utilisent des bibliothèques logicielles, comme OpenMP ou MPI (Quinn, 2004) permettant de simplifier leur parallélisation. Cependant, elles se caractérisent par un fort indéterminisme inhérent au paradigme de la programmation parallèle (condition de concurrences d'accès aux ressources, ordre des messages, interblocages), évoqué par Chassin de Kergommeaux et al. (2001), et pour lesquelles les méthodes d'analyse traditionnelles ne sont pas suffisantes.

Parmi les solutions les plus fréquemment employées pour l'analyse d'un système informatique figure le traçage. Tracer consiste à stocker, sous forme de fichiers, le résultat de l'exécution de l'application en enregistrant des informations sur l'état de l'exécution et des mesures sur l'état de la plate-forme issues de la récupération de valeurs contenues dans des compteurs matériels ou fournies par l'applicatif. Pour ce faire, le code de l'application est instrumenté à l'aide de bibliothèques logicielles, d'outils de traçage, ou simplement de fonctions d'affichage ou d'écriture dans des fichiers. Les éléments à tracer (fonction appelée, état d'une entité, commutation de contexte, interruption, valeur d'une variable, événement particulier) sont déterminés par l'analyste ou par l'outil. La trace est alors générée au fur et à mesure de l'exécution, sous la forme d'un format particulier (générique ou standardisé, textuel ou binaire selon les outils utilisés). Parmi les formats de traces les plus répandus, on peut citer OTF (Knüpfer et al., 2006) utilisé par des outils de visualisation de traces comme Vampir (Knüpfer et al., 2008), le format de trace de Tau (Shende, 2006), celui de l'outil de visualisation Pajé (Chassin de Kergommeaux, 2000), ou CTF, spécifié par Desnoyers (2012). Le traçage concerne aussi bien le monde des architectures largement distribuées, avec notamment des infrastructures comme Tau ou des outils comme Score-P (Score-P Project, 2012), que celui des systèmes de plus petite échelle (ordinateurs personnels, systèmes embarqués). On citera par exemple LTTng, patch pour Linux permettant de tracer les événements au niveau noyau et *user-space* (Toupin, 2011; Fournier et al., 2009), ou KPTrace (Prada-Rojas et al., 2009), proposé par STMicroelectronics pour le débogage de ses plate-formes.

Les principales difficultés dues au traçage sont posées par le volume des fichiers obtenus, lié à la durée d'exécution et au nombre d'événements. Cela rend fastidieux le suivi de l'évolution de l'application au cours du temps, nécessaire à cause de son indéterminisme. Différentes méthodes de traitement (filtrage, élimination du bruit, agrégation, fouille de données, reconnaissance de motifs) ont pour but de simplifier la trace et de la structurer, tandis que d'autres (intégration dans une base de données) permettent une gestion optimisée de son stockage. En bout de chaine, des représentations visuelles de la trace sont fréquemment employées. L'émergence de ces outils date des années 1970, pour faire face aux problématiques liées à l'analyse des systèmes parallèles, et ont vu jusqu'à nos jours l'intégration progressive de différentes techniques de visualisation provenant de domaines variés comme les statistiques, la gestion de projet ou encore le stockage de données. Le choix de ces méthodes repose sur la pertinence qu'elles apportent à l'analyste quant à sa compréhension de l'application, aussi bien au niveau de l'évolution de son comportement au cours du temps que de sa structure, et des relations qui lient ses différents composants.

Nous nous proposons, dans ce papier, de faire un état de l'art des méthodes de visualisations appliquées pour l'analyse de traces. Nous suivrons une progression dans le détail qui consiste à commencer par une analyse des caractéristiques générales de l'application à l'aide d'une synthèse globale s'appuyant sur des visualisations statistiques. Elle se poursuit par une analyse détaillée de l'exécution de la trace, à travers des représentations qui font intervenir un axe temporel et montrent les séquences d'événements se produisant dans la trace, mais aussi à travers des représentations structurelles, favorisant l'observation des interactions entre les composants de l'application, leur hiérarchie et leur topologie. Nous évoquerons ensuite les solutions proposées, à travers des extensions des techniques précédentes, afin de permettre le passage à l'échelle de l'analyse. Pour finir, nous conclurons par une synthèse des concepts évoqués, puis par des perspectives en vue d'étendre les fonctionnalités de ces procédés.

2 Synthèse globale

La synthèse globale consiste en une représentation d'informations contenues dans la trace traitées par des opérateurs statistiques. Leur utilisation est une approche pertinente pour débuter une analyse, selon la méthodologie proposée par Shneiderman (1996), par exemple (*overview, zoom and filter, then details on demand*). Parmi les représentations possibles, les formes textuelles sont couramment utilisées, mais des méthodes visuelles faisant appel à des diagrammes, graphiques, sont aussi employées, grâce à la facilité de lecture qu'elles procurent.

2.1 Visualisations statistiques

Graphiques à courbes en deux dimensions Il s'agit d'une représentation simple, constituée par deux axes et une courbe, traduisant la variation de la valeur d'une variable en fonction de la valeur d'un de ses paramètres. L'outil PerfExplorer (Huck et Malony, 2005) offre la possibilité de comparer l'efficacité relative ou le *speed-up* en fonction du nombre de processeurs grâce à ce type de représentations.

Histogrammes, diagrammes circulaires Les histogrammes et diagrammes circulaires sont des représentations statistiques classiques traditionnellement utilisées pour comparer la répartition d'un certain nombre de valeurs. On peut ainsi représenter, par exemple, le nombre de messages reçus par les processus, le taux de mémoire utilisé par les différentes machines. On trouve ce genre de visualisations à partir de ParaGraph (Heath et Etheridge, 1991). Paradyn (Miller et al., 1995) implémente des histogrammes horizontaux contenant plusieurs types de valeurs, chacun possédant des échelles différentes. Les diagrammes circulaires sont présents dans Pajé et permettent de quantifier le temps passé dans les différents états par chaque processus. La possibilité est laissée à l'utilisateur de définir la tranche de temps sur laquelle faire l'analyse. Cette méthode est idéale pour comparer deux processus et isoler des problèmes de performance reliés à un déséquilibre de répartition des tâches, par exemple.

Nuages de points Le nuage de points est une représentation graphique donnant des indications sur le degré de corrélation entre deux ou plusieurs variables liées. Elle permet ainsi d'observer des relations (directes, inverses), des dépendances (fortes ou faibles), des tendances (linéaires, non linéaires) entre les variables, d'avoir un aperçu de l'homogénéité des répartitions ou d'isoler des données aberrantes. Une visualisation de ce type est disponible au sein de l'outil PerfExplorer. Les auteurs se servent de la corrélation pour ignorer des données qui fourniraient des informations redondantes (comme celles fournies par certains compteurs matériels), en vue de simplifier l'analyse.

"Area charts" Ils consistent en une représentation en deux dimensions, basée sur un graphique à courbes, où sont superposées différentes quantités à observer, mises en exergue par des zones de couleurs différentes. L'intérêt est de pouvoir comparer la taille des aires respectives et leur évolution en fonction de l'abscisse. Dans PerfExplorer (figure 1) , cette représentation peut être utilisée afin de comparer l'évolution du temps relatif passé dans différentes fonctions ou états en fonction du nombre de processeurs.

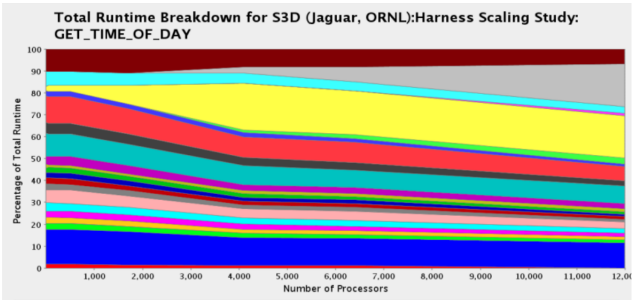

FIG. 1 – *Exemple de représentation statistique issue de PerfExplorer : un "area chart" représentant le pourcentage de temps passé dans différentes fonctions en fonction du nombre de processeurs.*

Représentations en trois dimensions Des représentations en trois dimensions sans axe temporel permettent d'élargir l'analyse en affichant une donnée en fonction de trois variables. ParaProf (Bell et al., 2003) possède ainsi ce type de visualisations pour des nuages de points, des histogrammes, ou à d'autres représentations à base de réseaux de triangles.

2.2 Conclusion sur les outils statistiques

Les visualisations synthétiques basées sur les statistiques donnent à l'analyste une première approche efficace de la trace mais ne suffisent pas à elles seules à permettre une rétro-action sur le code de l'application. Les informations fournies permettent de déceler un comportement général problématique (un processus gourmand en ressources, par exemple) mais les relations de causalité liant une configuration d'événements et d'états et leurs conséquences sur le déroulement de l'application ne sont pas explicités (comme des échéances non respectées, des interblocages). De même, les liens entre les composants de l'application ou leur structure topologique ne sont pas représentés, ce qui rend par exemple difficile l'analyse des communications afin de mettre en évidence des goulets d'étranglement sur le réseau. Les représentations statistiques incitent donc l'utilisateur à affiner son examen de la trace à travers des méthodes de visualisation permettant une analyse temporelle et structurelle détaillée de celle-ci, tout en leur fournissant des opérateurs mathématiques pertinents pour la gestion du passage à l'échelle.

3 Visualisation détaillée de la trace

Les techniques de visualisation présentes au sein des outils d'analyse donnent des informations détaillées sur le comportement de l'application mais aussi sur la structure de la trace. Les plus plébiscitées sont issues de domaines divers, comme la gestion de projet (diagramme de Gantt), le stockage de données (visualisation treemap), l'algèbre (représentations matricielles) ou l'analyse mathématique (graphes d'appels de fonctions). Un certain nombre d'entre-elles ont été catégorisées par Schnorr (2009) durant sa thèse, que nous avons complété en suivant la classification qu'il propose.

3.1 Visualisation du déroulement de l'exécution

Un certain nombre de techniques de visualisation permettent d'étudier le comportement d'une application au cours du temps à travers l'analyse de la séquence des événements, états et modification de la valeur de ses variables. L'objectif est en particulier de déduire les relations de causalité entre les différents événements aboutissant à un état particulier de l'application à un moment donné, pour éventuellement influer sur l'origine d'un comportement indésirable à partir de ces indications (redimensionnement de la plate-forme, modification du code source).

Diagrammes de Gantt Le diagramme de Gantt (Wilson, 2003) est apparu en 1896 et a d'abord été employé dans la gestion de projets. Dans le cas des outils de visualisation de trace, il est composé d'un axe temporel en abscisse tandis que l'axe des ordonnées correspond à l'ensemble des *conteneurs* (par exemple cœurs, processus, threads, fonctions) dont les états vont être représentés au cours du temps. Ces derniers sont disponibles sous la forme de rectangles disposés horizontalement, parfois colorés, dont les extrémités correspondent aux dates de début et de fin. Dans le cas d'événements ponctuels, certains outils se servent de symboles. KPTrace utilise, par exemple, des images de verrous pour indiquer la prise ou la libération d'un mutex ou d'une sémaphore tandis qu'il souligne un changement de contexte ou une préemption par des flèches verticales colorées et orientées. LTTng Eclipse viewer (Linux Tools Project, 2012) possède deux diagrammes de Gantt séparés représentant d'un côté les éléments logiciels (processus et fonctions) et de l'autre les entités matérielles (processeurs, IRQ). A l'inverse, Pajé, utilisant un format de trace générique, laisse la possibilité d'entrelacer la hiérarchie matérielle et logicielle.

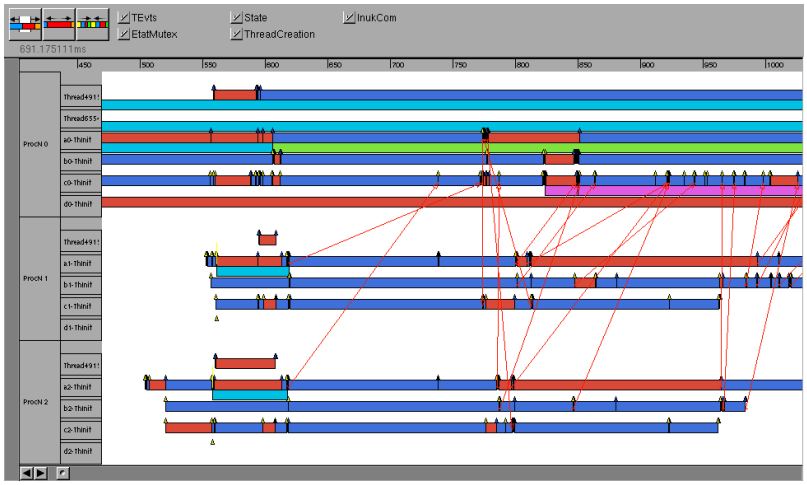

FIG. 2 – *Exemple de diagramme de Gantt issu de l'outil Pajé et montrant l'exécution de trois processus (états et communications)*

Graphiques à courbes, histogrammes, "area charts", avec axe temporel Représentations similaires à celles présentées dans la section 2.1, elles se distinguent ici par la présence d'un axe temporel en abscisse servant à montrer l'évolution de la valeur d'une ou plusieurs variables au cours du temps. Ces variables peuvent être des informations relatives au matériel : la Streamline de l'outil ARM DS5 (ARM Ltd., 2012) possède des représentations de ce type, sous la forme d'histogrammes (utilisation du processeur par le système et l'utilisateur), ou d'*area charts* (comparant deux à deux les interruptions logicielles et matérielles, les *cache hits* et *misses*, les écritures et lectures sur disque, la mémoire utilisée et disponible). A contrario, on peut aussi trouver des informations issues du logiciel : dans Pajé, il est ainsi possible de représenter des variables sous la forme d'une courbe intégrée au diagramme de Gantt principal. Ces techniques peuvent être transposées en trois dimensions, comme la *3D Terrain Visualization* de Paradyn, où les données représentées sont dépendantes de deux variables et du temps.

"3D View" Dans les visualisations comportementales, un axe est utilisé pour le temps. Cela oblige à projeter les ressources sur une dimension, au détriment de la compréhension de leur structure. Une solution proposée par l'auteur de Triva (Schnorr, 2009) est la *3D view* qui combine la structure, à travers une représentation de la topologie, et un axe temporel.

3.2 Visualisation de la structure de l'application

Les techniques de visualisation structurelles se caractérisent par la représentation des composants constituant l'application ainsi que leurs interactions. La dimension temporelle, si elle est présente, n'est pas centrale, comme dans les représentations évoquées dans la section 3.1.

Appels de fonctions Les graphes d'appels de fonction sont des graphes orientés, dans lesquels sont représentés les appels (symbolisés par des liens) entre les différents composants (fonctions ou méthodes représentés par des nœuds). Cette technique de visualisation est efficace pour l'analyse d'applications parallèles, en particulier celles conçues sous la forme d'un graphe de flot de données. Virtue (Shaffer et Reed, 2000) contient un graphe d'appel de fonctions intégré au sein d'un environnement en trois dimensions, permettant à l'utilisateur de manipuler la représentation (rotation, sélection de parties) afin d'obtenir plus d'informations sur les éléments affichés. L'outil Extravis (Cornelissen et al., 2008) propose une représentation circulaire (figure 3) sur le bord de laquelle les différentes fonctions sont indiquées. Un gradient de couleur appliqué sur les liens donne au choix des informations sur le sens des appels, ou sur leur survenue dans le temps.

Communications Les communications entre différents processus à un instant donné peuvent être matérialisées par des matrices. Ces dernières consistent en une représentation à deux dimensions, sous la forme d'une grille. Des couleurs peuvent être employées pour indiquer le type de communications ou la taille des données transmises. Elles sont un outil idéal pour des applications parallèles en permettant la détection de goulets d'étranglement, et figurent dans le choix des visualisations proposées par Vampir. Une méthode alternative pour représenter les communications est le graphe orienté. Les entités sont symbolisées par les nœuds, tandis que les liens indiquent les communications. On en trouve une implémentation dans ParaGraph. Le graphe topologique en est une version étendue où la position relative des entités représente la

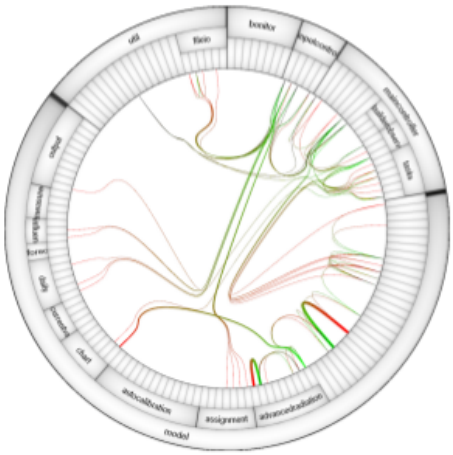

FIG. 3 – *Graphe d'appels de fonctions circulaire d'Extravis. Les liens symbolisent les appels entre les fonctions, le gradient de couleur indiquant leur sens.*

topologie réelle du réseau. Implémenté dans Triva (Schnorr et al., 2011), il possède un mécanisme d'animation temporel qui permet d'analyser au cours du temps la variation des volumes de communications.

Topologie de la plate-forme La hiérarchie des ressources et la topologie d'une plate-forme peuvent être représentés à l'aide d'arbres. Une technique d'arbres particulière est la représentation *treemap* inventée par Shneiderman (1991). Elle consiste à diviser l'espace de visualisation (l'écran) en représentant chacune des entités sous la forme d'un rectangle, dont la surface est proportionnelle à la valeur de la variable que l'on veut mettre en évidence. La treemap est particulièrement utile pour représenter un grand nombre d'entités grâce à la surface disponible qu'elle offre et la possibilité de représenter les niveaux hiérarchiques. Elle est employée dans Triva (Schnorr et al., 2012), dans sa version *squarified*, pour montrer la répartition des tâches sur de larges architectures distribuées. En particulier, les auteurs utilisent le même mécanisme d'animation temporel que pour les graphes topologiques afin de montrer la variation du ratio du temps attribué à l'exécution de deux tâches sur chaque entité, au cours du déroulement de l'application.

3.3 Problématiques liées aux méthodes de visualisation présentées

Les méthodes visuelles présentées ne répondent pas toutes de la même manière à des contraintes comme la gestion de l'échelle (nombre important d'entités produisant les événements, hiérarchie complexe) ou la diversité et de la quantité des informations à analyser (durée de la trace, nombre d'événements, hétérogénéité des durées des événements). Des symptômes comme les pertes de contexte, dues à la nécessité de *scroller* ou dézoomer (par exemple dans le cas du diagramme de Gantt, pour afficher l'intégralité des conteneurs ou se déplacer dans

l'axe du temps), nuisent à la compréhension générale de la trace. En outre, un déplacement dans la trace ou un changement d'échelle sont aussi associés à des phénomènes de crénelage : si la taille d'un élément à afficher après un zoom ou un dézoom est un nombre réel non entier, il ne pourra pas être représenté correctement à cause de la discrétisation de l'écran. Si cet objet est déplacé, la taille affichée sera potentiellement modifiée en fonction de sa position par rapport aux pixels de l'écran, alors qu'il ne sera pas pas censé avoir été redimensionné. Ces artefacts sont susceptibles d'induire l'analyste en erreur, voire de donner une représentation intégralement illisible dans le cas d'un fort dézoom. Certaines représentations dont la taille des éléments est liée à la surface d'affichage disponible (treemaps, matrices) peuvent voir ces dernières devenir trop petites à cause d'un nombre d'entités trop important.

4 Passage à l'échelle de l'analyse

Afin de palier au manque de surface disponible pour afficher les visualisations et répondre aux problématiques liées au passage à l'échelle, différents auteurs ont mis au point des solutions basées sur des mécanismes d'agrégation. La méthodologie employée nécessite de définir trois critères :

- Le choix de la dimension d'agrégation, corrélée avec la technique de visualisation qui va servir de support. On distingue en particulier la dimension temporelle et la dimension spatiale. Cette dernière se fonde sur des critères structurels de l'application liés à l'ensemble de ses ressources.
- Le choix de la métrique de proximité, qui est une condition ou un critère dépendant de l'axe et qui va déclencher l'agrégation. Cette métrique permet de déterminer l'ensemble des éléments sur lesquels effectuer l'opération.
- Le choix de l'opérateur qui va être appliqué sur l'ensemble des données à agréger et ainsi fournir une représentation de cet agrégat.

Exemples d'agrégations Parmi les outils employant des agrégations basées sur l'axe temporel, on peut citer LTTng et Pajé (diagrammes de Gantt), mais aussi Triva (animation temporelle). La métrique de proximité de LTTng est une taille de pixel minimale liée à la représentation des états. Lors d'un dézoom, si la taille de l'objet devient plus petite que le seuil toléré, alors il est visuellement agrégé, ce qui est matérialisé par un point au-dessus de sa position. On retrouve un principe similaire dans Pajé, où la métrique est un groupe d'états contigus. Si l'un d'eux possède, suite à un redimensionnement, une taille inférieure à un certain nombre de pixels, le groupe d'états est agrégé sous la forme d'un état unique barré d'une ligne diagonale. Dans le cas de Triva, la métrique est une tranche de temps, dont l'analyste peut paramétrer la durée, et selon laquelle la trace va être découpée. Tous les événements contenus dans chaque tranche de temps seront alors agrégés, en calculant la moyenne des durées des états ou des valeurs des variables au cours de la tranche de temps, ou la somme des communications ou des événements ponctuels.

Dans le cas d'une agrégation selon l'axe spatial, le diagramme KPTrace permet d'agréger les ressources selon une métrique déterminée par la proximité hiérarchique : tous les conteneurs fils sont masqués et rassemblés au sein de leur conteneur père. Les états et événements associés aux fils sont alors déportés dans la ligne correspondant au père. Pajé permet aussi la même agrégation hiérarchique, bien que les états agrégés correspondant aux fils ne soient

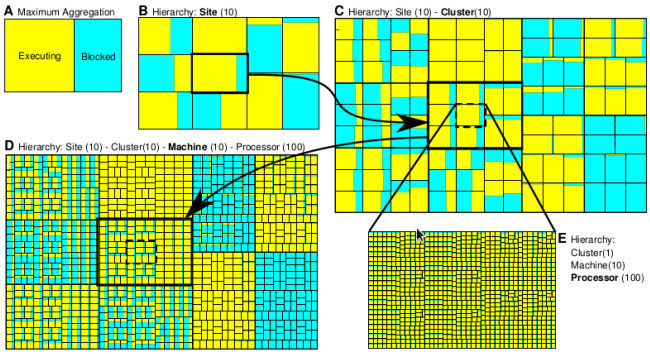

FIG. 4 – *Agrégation hiérarchique de la treemap proposée par Triva. On se déplace progressivement au sein de la hiérarchie en désagrégeant et en se focalisant sur une partie de la représentation.*

représentés que sous la forme d'un état indéterminé (les communications avec des conteneurs externes sont cependant toujours affichées). La même métrique de proximité spatiale est utilisée par Triva au sein de la treemap (figure 4) et du graphe topologique. Les fils ne sont plus représentés, et les valeurs caractérisant le père sont obtenues à l'aide de l'opérateur *somme* appliqué sur l'ensemble des fils. Dans le cas du regroupement par profil proposé par Vampir, la métrique est liée à une proximité "comportementale" : les différentes tâches qui possèdent un profil d'exécution proche, dont la similitude est évaluée à l'aide d'une distance euclidienne appliquée sur un jeu de paramètres (fonctions, durées de celles-ci), sont rassemblées. L'ensemble des profils est ensuite affiché sous la forme d'un histogramme. La durée des états contenu dans chaque profil est la moyenne de celles des processus agrégés.

4.1 Problématiques liées aux méthodes d'agrégation proposées

Les solutions étudiées, si elles possèdent un certain nombre de techniques permettant de répondre aux besoins de passage à l'échelle, amènent toutefois à certains questionnements.

Pertes d'informations liées à l'agrégation de données Les opérateurs utilisés afin d'agréger les données contenues dans la trace font irrémédiablement perdre une certaine quantité d'information : la moyenne efface celles concernant la distribution des données, tandis que des opérateurs de distribution ne donnent des informations que sur une seule des valeurs du groupe agrégé. Dans le cas de l'agrégation temporelle de Triva, par exemple, la valeur moyenne calculée sur tranche de temps ne permet pas de déterminer le nombre d'occurrences de chaque état, leur périodicité, ou encore la variance de leur durée. Pour une analyse détaillée de la distribution et de la régularité, cette agrégation sera alors insuffisante. De même, des méthodes d'agrégation visuelles, comme celles proposées par LTTng Viewer ou Pajé, permettent de savoir qu'un état agrégé est "trop petit" pour être représenté, mais si on compare deux états agrégés, il n'y a plus d'informations sur leur durée relative. La perte de l'information tolérée est reliée à la pertinence de la technique employée pour le type de comportement à visualiser :

un profilage utilisant le regroupement de profil de Vampir se satisfera d'une perte d'information due à l'agrégation, à contrario de l'analyse des interblocages qui aura besoin de plus de précision. Detyniecki (2001) évoque un certain nombre d'opérateurs d'agrégations dont l'utilisation pourrait être envisagée afin de mieux contrôler la perte d'information, mais aussi afin de faire ressortir des caractéristiques comportementales que les opérateurs les plus simples ne permettent pas de mettre en évidence. De même, des efforts peuvent être poursuivis afin d'attribuer plus de sémantique aux agrégats visuels (taille des symboles corrélée à une information perdue par l'agrégation, par exemple).

Phénomènes d'instabilité On peut parler d'instabilité si une légère variation des paramètres d'agrégation peut générer une représentation différente. Cela peut être visible lors d'une agrégation temporelle utilisant une tranche de temps : lors d'un redimensionnement de ces dernières, la distribution des événements va changer, et certains vont passer d'une tranche à l'autre, être répartis sur plusieurs tranches de temps, ou passer d'un état réparti à une tranche unique. Ainsi, pour deux représentations possédant n et $n+1$ tranches de temps, l'amplitude associée aux tranches de temps "correspondantes", de par leur proximité temporelle, pourra être différente. Le phénomène produira deux représentations agrégées différentes et visuellement non cohérentes malgré des paramètres d'agrégation très proches. Pour rétablir la cohérence, on pourrait ajouter des contraintes sur la métrique de proximité (tranches de temps multiples les unes des autres) ou envisager un lissage ou un filtrage, avec toutefois le risque de modifications ou de pertes d'informations.

5 Conclusion

Les méthodes de visualisation statistiques, temporelles et structurelles de traces d'exécution sont des techniques complémentaires répondant en partie aux besoins liés à l'analyse des systèmes informatiques actuels. Elles fournissent des informations générales sur la trace, sur le comportement des applications au cours du temps et les relations entre les différents composants de la plate-forme étudiée. Face aux problématiques de passage à l'échelle dues à des architectures structurellement complexes et un niveau de détail important, des solutions faisant appel à des mécanismes d'agrégation simplifient la visualisation en agissant au niveau des dimensions temporelles et spatiales. Cependant, ces techniques sont limitées par la perte d'informations qu'elles induisent, la fiabilité des représentations qu'elles produisent, ou à cause d'effets d'instabilités. De plus, la pertinence de leur utilisation est fortement corrélée à la structure de la plate-forme, à l'application à analyser, et au type de comportements à discriminer.

Les outils actuels ne laissent pas le choix des opérateurs, de l'ensemble des données ou de la partie de la trace sur lesquels agir. Des agrégations successives, combinées, ou la représentation d'un même objet en utilisant simultanément des agrégations différentes n'ont pas encore été évoquées dans la littérature concernant l'analyse visuelle des traces. Nous pensons qu'étendre les possibilités offertes par ce procédé, à travers l'interaction et la possibilité d'implanter de nouvelles fonctions d'agrégation, est une solution face aux limitations dont il souffre actuellement. En particulier, la multiplication des possibilités d'agrégation serait une solution remédiant aux pertes d'informations en permettant le recoupage des données.

Références

ARM Ltd. (2012). ARM streamline performance analyzer - ARM. http ://www.arm.com/products/tools/software-tools/ds-5/streamline.php.

Bell, R., A. D. Malony, et S. Shende (2003). ParaProf : a portable, extensible, and scalable tool for parallel performance profile analysis. In *Euro-Par 2003 Parallel Processing*, Volume 2790, pp. 17–26. Berlin, Heidelberg : Springer Berlin Heidelberg.

Campbell, A., G. Coulson, et M. Kounavis (1999). Managing complexity : middleware explained. *IT Professional 1*(5), 22 –28.

Chassin de Kergommeaux, J. (2000). Pajé, an interactive visualization tool for tuning multithreaded parallel applications. *Parallel Computing 26*(10), 1253–1274.

Chassin de Kergommeaux, J., Ã. Maillet, et J.-M. Vincent (2001). *Program Development for Cluster Computing : Methodology, Tools and Integrated Environments*, Chapter Monitoring Parallel Programs for Performance Tuning in Distributed Environments, chapter 6. Nova Science.

Cornelissen, B., A. Zaidman, D. Holten, L. Moonen, A. Van Deursen, et J. van Wijk (2008). Execution trace analysis through massive sequence and circular bundle views. *Journal of Systems and Software 81*(12), 2252–2268.

Desnoyers, M. (2012). Common trace format (CTF) specification (v1.8.2). http ://git.efficios.com/ ?p=ctf.git ;a=blob_plain ;f=common-trace-format-specification.txt.

Detyniecki, M. (2001). Fundamentals on aggregation operators. *This manuscript is based on Detyniecki's doctoral thesis and can be downloaded from*.

Fournier, P. M., M. Desnoyers, et M. R. Dagenais (2009). Combined tracing of the kernel and applications with LTTng. In *Proceedings of the 2009 Linux Symposium*.

Heath, M. T. et J. A. Etheridge (1991). Visualizing the performance of parallel programs. *IEEE Software 8*(5), 29–39.

Herlihy, M. et N. Shavit (2008). *The art of multiprocessor programming*. Amsterdam ; London : Elsevier/Morgan Kaufmann.

Huck, K. A. et A. D. Malony (2005). Perfexplorer : A performance data mining framework for large-scale parallel computing. In *Proceedings of the 2005 ACM/IEEE conference on Supercomputing*, pp. 41.

Hwang, K., J. J. Dongarra, et G. C. Fox (2012). *Distributed and cloud computing*. Amsterdam ; London : Elsevier/Morgan Kaufmann.

Knüpfer, A., R. Brendel, H. Brunst, H. Mix, et W. E. Nagel (2006). Introducing the open trace format (OTF). In *Computational Science – ICCS 2006*, Volume 3992, pp. 526–533. Berlin, Heidelberg : Springer Berlin Heidelberg.

Knüpfer, A., H. Brunst, J. Doleschal, M. Jurenz, M. Lieber, H. Mickler, M. S. Müller, et W. E. Nagel (2008). The vampir performance analysis tool-set. In *Tools for High Performance Computing*, pp. 139–155. Berlin, Heidelberg : Springer Berlin Heidelberg.

Linux Tools Project (2012). Linux tools Project/LTTng2/User guide - eclipsepedia. http ://wiki.eclipse.org/index.php/ Linux_Tools_Project/LTTng2/User_Guide.

Miller, B. P., M. D. Callaghan, J. M. Cargille, J. K. Hollingsworth, R. B. Irvin, K. L. Karavanic,

K. Kunchithapadam, et T. Newhall (1995). The paradyn parallel performance measurement tool. *Computer 28*(11), 37–46.

Prada-Rojas, C., F. Riss, X. Raynaud, S. De-Paoli, et M. Santana (2009). *Observation tools for debugging and performance analysis of embedded linux applications*. September.

Quinn, M. J. (2004). *Parallel programming in C with MPI and OpenMP*. Boston [etc.] : McGraw-Hill.

Schnorr, L. M. (2009). *Quelques Modèles de Visualisation pour l'Analyse des Applications Parallèles*. Ph. D. thesis.

Schnorr, L. M., G. Huard, et P. O. A. Navaux (2012). A hierarchical aggregation model to achieve visualization scalability in the analysis of parallel applications. *Parallel Computing 38*(3), 91–110.

Schnorr, L. M., A. Legrand, et J.-M. Vincent (2011). Detection and analysis of resource usage anomalies in large distributed systems through multi-scale visualization. *Concurrency and Computation : Practice and Experience*, n/a–n/a.

Score-P Project (2012). Score-P – HPC profiling and event tracing infrastructure. http ://www.vi-hps.org/projects/score-p.

Shaffer, E. et D. Reed (2000). Real-time immersive performance visualization and steering. *ACM SIGGRAPH Computer Graphics 34*(2), 11–14.

Shende, S. S. (2006). The tau parallel performance system. *International Journal of High Performance Computing Applications 20*(2), 287–311.

Shneiderman, B. (1991). Tree visualization with tree-maps : A 2-d space-filling approach. *ACM Transactions on Graphics 11*, 92–99.

Shneiderman, B. (1996). The eyes have it : A task by data type taxonomy for information visualizations. In *Visual Languages, 1996. Proceedings., IEEE Symposium on*, pp. 336–343.

Toupin, D. (2011). Using tracing to diagnose or monitor systems. *Software, IEEE 28*(1), 87–91.

Wilson, J. (2003). Gantt charts : A centenary appreciation. *European Journal of Operational Research 149*(2), 430–437.

Summary

Behavior analysis of an application software is a difficult task because of computing system growing complexity. This phenomenon is characterized by complex sofware stacks in embedded and personnal systems case, while parallel systems analysis has to deal with large architectures and strong indeterminism. Execution traces visualization usage related to software analysis has rised because of its ability to solve these issues. Analysis tools propose now a large set of techniques, distinguished by data volume, system scale, or type of software behavior they can represent. We propose to study them accross this survey by discussing about statistics methods, then about behavioral and structural visualisations which give precise details on the applications. We will extend our explanations to techniques capable to solve the issues related to scalability found in most of traditional visualization ways.