Agrégation temporelle pour l'analyse de traces volumineuses

Damien Dosimont[3,1,2], Guillaume Huard[1,3,2], Jean-Marc Vincent[1,3,2]

[1] U. Grenoble Alpes, LIG, 38000 Grenoble, France
[2] CNRS, LIG, F-38000 Grenoble, France
[3] Inria
prénom.nom@imag.fr

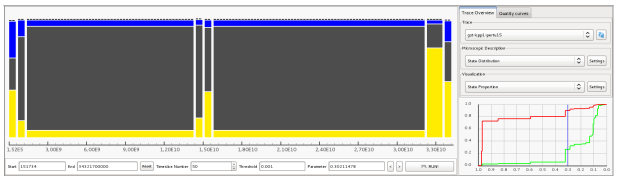

FIGURE 1 – L'outil Ocelotl montrant les différentes phases d'un décodage vidéo avec GStreamer. Une perturbation aux alentours de $1,5 \cdot 10^{10}$ ns est mise en évidence. Les courbes représentant la perte d'information et la réduction de complexité forment l'interface de choix du compromis d'agrégation.

1. Visualisation de traces volumineuses

L'analyse de la trace d'exécution d'une application est difficile quand la quantité d'événements qu'elle contient est importante. Les principales limites sont dues à la surface d'écran disponible, en particulier lors de l'utilisation de techniques représentant les ressources et le temps. Le diagramme de Gantt, employé par les analystes pour comprendre les relations de causalité et la structure de l'application, en est un exemple.

Différentes approches tentent de résoudre ces problèmes liés au passage à l'échelle de l'analyse. L'agrégation visuelle agrège les objets graphiques trop petits pour être affichés [1] ou adapte le rendu aux pixels disponibles [2]. Toutefois, ces approches ne représentent pas d'information utile à l'analyste dans le premier cas et sont instables en cas de redimensionnement dans le second.

L'agrégation d'information s'attaque au problème en amont, en diminuant la complexité des données à afficher. Viva [6] représente les ressources sous forme de treemap et propose une agrégation temporelle de leurs valeurs, mais le temps n'est pas représenté explicitement. Jumpshot [4] fournit, quant à lui, différent niveaux d'abstraction temporels de la trace mais ne permet pas d'agréger les ressources.

Dans le but de fournir une vue d'ensemble temporelle d'une trace, ce que ne fournissent pas les techniques actuelles, nous proposons une nouvelle technique, implémentée dans l'outil Ocelotl (Figure 1). Cette technique permet une analyse temporelle macroscopique qui n'est pas gênée par l'affichage d'un grand nombre de ressources. Elle représente le déroulement de l'application au cours du temps en agrégeant les parties de la trace où le comportement des ressources est homogène. Cette agrégation est modulée dynamiquement par l'utilisateur qui choisit un compromis entre la complexité et la perte d'information.

2. Représentation macroscopique de la trace

Notre contribution se décompose en deux points. Nous avons d'abord généralisé un algorithme d'agrégation temporelle [3], destiné à l'analyse macroscopique des grands systèmes. Un système est d'abord décrit sous la forme d'un modèle microscopique, associé à des métriques permettant d'interpréter son comportement. Le processus d'agrégation permet ensuite de représenter ce système sous la forme d'une partition du modèle microscopique. Cette partition est obtenue grâce à un compromis, dont l'utilisateur a le contrôle, entre la complexité (liée au nombre d'agrégats représentés) et la quantité d'information perdue en agrégeant. L'algorithme aura donc tendance à agréger les valeurs les plus proches en priorité afin de minimiser cette perte d'information pour une complexité donnée. Dans notre cas, nous n'autorisons l'agrégation qu'entre zones temporelles contigües. Notre première contribution a été de définir le modèle microscopique représentant le comportement de l'application, en découpant la trace en tranches de temps fixes pour lesquelles sont calculées la distribution des états des ressources. Nous avons aussi dû étendre l'algorithme initial à des quantités multidimensionnelles afin de pouvoir agréger le modèle microscopique que nous proposons. La représentation de la trace est faite avec une simple ligne qui montre le découpage en une succession temporelle d'agrégats. Chacun de ces agrégats représente un comportement particulier dans la trace. En outre, nous représentons pour chaque agrégat la proportion des différents états qu'il contient.

Notre seconde contribution est l'outil Ocelotl dans lequel la technique est implémentée. En tirant parti de l'infrastructure d'analyse de traces FrameSoC [5], dans laquelle il vient se greffer, Ocelotl peut afficher des traces de plusieurs Go et de dizaines de millions d'événements dans des délais raisonnables. A titre d'exemple, une trace de 2.3 Go et 15 millions

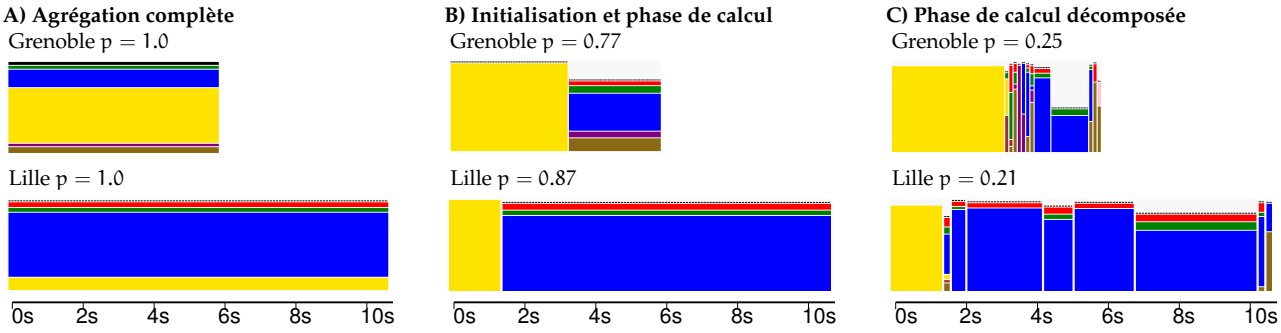

FIGURE 2 – Analyse comparative du benchmark NAS LU.A sur les sites de Grenoble et de Lille. Trois compromis distincts sont représentés. Les couleurs représentent la proportion des différents états MPI pour chaque agrégat ((`MPI_Recv` en bleu, `MPI_Wait` en rouge et `MPI_Send` en vert).

d'événements demande 5 mn de prétraitement sur un PC à 4 cœurs Intel Core i7-2760QM 2.4 GHz et 8 Go de DDRAM. Cela est possible grâce à l'utilisation de bases de données relationnelles pour stocker les traces, ce qui permet des requêtes optimisées afin d'éviter de saturer la mémoire. Ocelotl fournit des interactions pour zoomer, dézoomer et changer dynamiquement le niveau de détail de l'agrégation avec des temps de réaction immédiats. Il permet aussi de commuter vers un diagramme de Gantt sur une zone temporelle particulière, afin d'obtenir plus de détails.

3. Analyse de différents types de systèmes

Notre technique a été validée par l'analyse de plusieurs cas d'études. Notre premier cas est la lecture d'une vidéo avec GStreamer, perturbée par un stress CPU artificiel. La figure 1 montre la détection de cette perturbation dans une vidéo de 34 s.
La figure 2 représente l'exécution d'une application MPI (NAS Benchmark) sur deux plate-formes de calcul équivalentes en terme de nombre de ressources (Grid'5000, sites de Grenoble, 160 cœurs et de Lille, 152 cœurs). L'intérêt est de faire apparaître l'influence de la plate-forme sur le déroulement de l'application (phases de calcul de tailles différentes, proportions et séquences des états MPI qui varient, en particulier lors des communications).

4. Perspectives

La technique d'agrégation temporelle que nous avons implémentée dans Ocelotl permet de représenter une grande quantité d'information sous la forme d'une description macroscopique simple dans des délais adaptés à une analyse interactive.
Nous aimerions, cependant, pouvoir représenter conjointement les ressources, en particulier dans les cas hétérogènes, afin de déterminer leur influence sur les performances de l'application. Nous sommes ainsi focalisés actuellement sur une extension de ces travaux à l'agrégation simultanée de l'espace des ressources et du temps.

Bibliographie

1. Jacques Chassin de Kergommeaux. Pajé, an Interactive Visualization Tool for Tuning Multi-Threaded Parallel Applications. *Parallel Computing*, 26(10) :1253–1274, August 2000.
2. Andreas Knüpfer, Holger Brunst, Jens Doleschal, Matthias Jurenz, Matthias Lieber, Holger Mickler, Matthias S. Müller, and Wolfgang E. Nagel. The Vampir Performance Analysis Tool-Set. In *Tools for High Performance Computing*, pages 139–155. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.
3. Robin Lamarche-Perrin, Yves Demazeau, and Jean-Marc Vincent. The Best-partitions Problem : How to Build Meaningful Aggregations. In *Proceedings of the 2013 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT'13)*, Atlanta, USA, 2013.
4. Ewing Lusk and Anthony Chan. Early experiments with the OpenMP/MPI hybrid programming model. In Rudolf Eigenmann and Bronis R. de Supinski, editors, *OpenMP in a New Era of Parallelism*, volume 5004 of *Lecture Notes in Computer Science*, pages 36–47. Springer, 2008. IWOMP, 2008.
5. Generoso Pagano, Damien Dosimont, Guillaume Huard, Vania Marangozova-Martin, and Jean-Marc Vincent. Trace Management and Analysis for Embedded Systems. In *Proceedings of the IEEE7th International Symposium on Embedded Multicore SoCs (MCSoC-13)*, Tokyo, Japan, sep 2013.
6. Lucas Mello Schnorr, Arnaud Legrand, and Jean-Marc Vincent. Detection and analysis of resource usage anomalies in large distributed systems through multi-scale visualization. *Concurrency and Computation : Practice and Experience*, 24(15) :1792–1816, 2012.