AGENCE NATIONALE DE LA RECHERCHE ANR	**Compte-rendu de fin de projet**	

Projet ANR-18-CE25-0005

Pythia : décisions à l'exécution se basant sur des prédictions

Programme JCJC 2018

A IDENTIFICATION

Acronyme du projet	Pythia
Titre du projet	Pythia : décisions à l'exécution se basant sur des prédictions
Coordinateur du projet (société/organisme)	François Trahay (Télécom SudParis)
Période du projet (date de début – date de fin)	01/03/2019 - 31/08/2022
Site web du projet, le cas échéant	https://github.com/trahay/ANR-Pythia

Rédacteur de ce rapport	
Civilité, prénom, nom	François Trahay
Téléphone	01 60 76 47 40
Adresse électronique	francois.trahay@telecom-sudparis.eu
Date de rédaction	31/08/2022

Liste des partenaires présents à la fin du projet (société/organisme et responsable scientifique)	Télécom SudParis

B RÉSUMÉ CONSOLIDÉ PUBLIC

B.1 RÉSUMÉ CONSOLIDÉ PUBLIC EN FRANÇAIS

Pythia : un oracle pour guider les décisions des supports d'exécution

Les supports d'exécution (en anglais *runtime*) prennent des décisions critiques pour les performances d'applications parallèles. Par exemple, le runtime OpenMP réparti une charge de travail entre plusieurs threads, ou le runtime MPI transmet des messages entre plusieurs processus s'exécutant sur une grappe de calcul. Malheureusement, ces décisions ne peuvent se baser que sur des heuristiques prenant en compte l'état actuel de l'application, en estimant son comportement probable dans le futur. Par conséquent, les supports d'exécution prennent parfois des décisions qui détériorent les performances au lieu de les améliorer.

Pythia vise à fournir aux supports d'exécution des moyens de prédire le futur de manière précise. Pour cela, Pythia se base sur la nature déterministe de nombreuses applications parallèles : généralement, un programme aura le même comportement d'une exécution à l'autre. Dans le cadre de Pythia, nous concevons une chaîne d'outils qui analyse l'exécution d'un programme afin de fournir aux supports d'exécution des indications lors d'exécutions futures du même programme. Grâce à des indications, un support d'exécution peut prendre des décisions en se basant à la fois sur l'état actuel de l'application, et sur le comportement futur du programme.

Analyse de traces d'exécution pour la prédiction

Nous proposons Pythia, une bibliothèque qui permet aux supports d'exécution de capturer une trace de l'exécution d'un programme. Pour cela, un support d'exécution indique certains événements (typiquement, le début ou la fin de certaines fonctions) à Pythia qui stocke les événements sous la forme d'une structure de données qui décrit le comportement de l'application. Cette structure de données contient les principales structures d'exécution du programme (boucles, branchements, séquences de fonction, etc.)

Lors des exécutions suivantes, Pythia charge la structure de données produite et compare l'exécution de l'application à la trace. Un support d'exécution peut demander à Pythia quels événements vont se produire dans le futur et quand ces événements se produiront. La prédiction produite par Pythia permet ensuite au support d'exécution de prendre une décision basée sur le comportement futur de l'application au lieu de s'appuyer sur une heuristique.

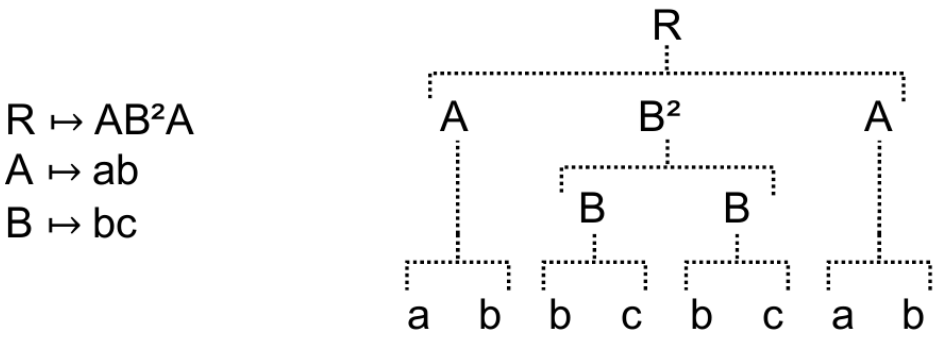
Résultats majeurs du projet

Nous avons développé la bibliothèque Pythia qui permet de capturer le comportement d'une application et de générer une trace d'exécution décrivant son déroulement. Les supports d'exécution peuvent demander à Pythia des prédictions sur le comportement futur de l'application afin de prendre les décisions les plus judicieuses. Ces travaux ouvrent la voie à de nouvelles optimisations dans des supports d'exécution : au lieu de reposer sur des heuristiques pour estimer le comportement probable d'une application, l'utilisation de prédiction permettra de connaître précisément l'avenir.

Production scientifique

Le projet Pythia a produit plusieurs logiciels open-source, notamment la bibliothèque Pythia, l'outil de collecte de trace EZTrace, ainsi que le logiciel d'analyse de traces EasyTraceAnalyser. Ces logiciels sont disponibles en ligne, et les détails de leur implémentation sont présentés sous la forme de plusieurs articles scientifiques publiés au cours du projet.

Illustration



Représentation d'une séquence d'événements sous la forme d'une grammaire.

Informations factuelles

Le projet Pythia est un projet de recherche coordonné par François Trahay (Télécom SudParis, laboratoire Samovar). Le projet a commencé en février 2019 et a duré 42 mois. Il a bénéficié d'une aide ANR de 184 056 €.

B.2 RÉSUMÉ CONSOLIDÉ PUBLIC EN ANGLAIS

Pythia: an oracle to guide runtime decisions

Runtime systems make decisions that are critical to the performance of parallel applications. For example, the OpenMP runtime distributes a workload among several threads, or the MPI runtime transfers messages between several processes running on a cluster. Unfortunately, these decisions can only be based on heuristics that take into account the current state of the

application, estimating its likely behavior in the future. As a result, runtime systems sometimes make decisions that degrade performance instead of improving it.

Pythia aims to provide runtime systems with the means to accurately predict the future. To do this, Pythia relies on the deterministic nature of many parallel applications: generally, a program will behave the same from one run to the next. In Pythia, we design a toolchain that analyzes the execution of a program in order to provide hints to runtime systems on future executions of the same program. With hints, a runtime system could make decisions based on both the current state of the application, and the future behavior of the program.

Execution trace analysis for prediction

We propose Pythia, a library that allows runtime systems to capture a trace of the execution of a program. To do this, a runtime system indicates certain events (typically, the start or end of certain functions) to Pythia which stores the events in the form of a data structure that describes the behavior of the application. This data structure contains the main execution structures of the program (loops, branches, function sequences, etc.)

On subsequent runs, Pythia loads the produced data structure and compares the execution of the application with the trace. An runtime system can ask Pythia what events will occur in the future and when these events will occur. The prediction produced by Pythia then allows the runtime system to make a decision based on the future behavior of the application instead of relying on a heuristic.

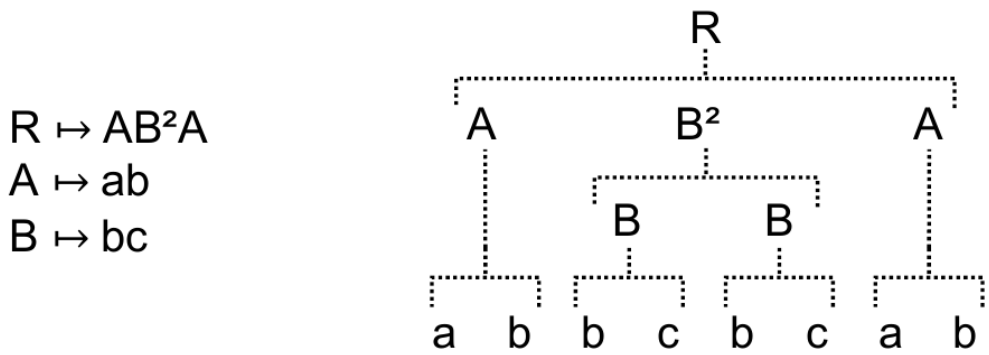
Main results of the project

We have developed the Pythia library which allows to capture the behavior of an application and to generate an execution trace describing its progress. Runtime systems can ask Pythia to make predictions about the future behavior of the application. This work opens the way to new optimizations in runtime systems: instead of relying on heuristics to estimate the likely behavior of an application, the use of prediction will allow to know precisely the future.

Scientific production

The Pythia project has produced several open-source software products, including the Pythia library, the EZTrace trace collection tool, and the EasyTraceAnalyser trace analysis software. These software products are available online, and the details of their implementation are presented in the form of several scientific papers published during the project.

Illustration



Representation of a sequence of events in the form of a grammar.

Facts and figures

The Pythia project is a research project coordinated by François Trahay (Télécom SudParis, Samovar laboratory). The project started in February 2019 and lasted 42 months. It received an ANR grant of €184,056.

C MÉMOIRE SCIENTIFIQUE

Mémoire scientifique confidentiel : non

C.1 RÉSUMÉ DU MÉMOIRE

Les supports d'exécution (en anglais *runtime*) prennent des décisions critiques pour les performances d'applications parallèles. Par exemple, le runtime OpenMP réparti une charge de travail entre plusieurs threads, ou le runtime MPI transmet des messages entre plusieurs processus s'exécutant sur une grappe de calcul. Malheureusement, ces décisions ne peuvent se baser que sur des heuristiques prenant en compte l'état actuel de l'application, en estimant son comportement probable dans le futur. Par conséquent, les supports d'exécution prennent parfois des décisions qui détériorent les performances au lieu de les améliorer.

Pythia vise à fournir aux supports d'exécution des moyens de prédire le futur de manière précise. Pour cela, Pythia se base sur la nature déterministe de nombreuses applications parallèles : généralement, un programme aura le même comportement d'une exécution à l'autre. Dans le cadre de Pythia, nous concevrons une chaîne d'outils qui analyse l'exécution d'un programme afin de fournir aux supports d'exécution des indications lors d'exécutions futures du même programme. Grâce à des indications, un support d'exécution peut prendre des décisions en se basant à la fois sur l'état actuel de l'application, et sur le comportement futur du programme.

C.2 ENJEUX ET PROBLÉMATIQUE, ÉTAT DE L'ART

Les supports d'exécution se chargent d'exploiter le matériel à la place du développeur du programme en fournissant une interface à l'application. La tâche consistant à exploiter efficacement un matériel particulier revient donc au *runtime*, et le développeur peut se concentrer sur l'algorithmique. Lors du choix de la stratégie à adopter, les supports d'exécution analysent le comportement passé de l'application afin d'estimer son comportement futur. Par exemple, certaines bibliothèques d'E/S analysent les demandes de lecture de données d'une application afin de précharger les blocs suivants, et ainsi réduire le temps d'attente lors des lectures suivantes [1], [2].

Cependant, le *runtime* ne fait que réagir à l'application au lieu d'anticiper son comportement. Cela entraîne de nombreux problèmes car certaines décisions précoces peuvent entrer en conflit avec des décisions ultérieures. Par exemple, sur une machine NUMA l'emplacement où la mémoire est initialement allouée a un impact significatif sur les accès mémoire ultérieurs et corriger le problème en déplaçant une page de mémoire plus proche d'un thread a un impact significatif sur les performances de l'application[3].

D'un autre côté, des années de recherche sur l'analyse de performance ont conduit à de nombreux outils qui sont capables d'analyser le comportement d'une application et d'identifier les causes probables de mauvaises performances [4]–[6]. Puisque les outils d'analyse des

performances collectent diverses informations pendant l'exécution d'un programme mais les analysent *post-mortem*, ils peuvent effectuer des traitements coûteux sans impacter les performances de l'application. À l'inverse, un support d'exécution doit agir rapidement afin de réduire son impact sur les performances du système. Ainsi, les outils d'analyse des performances peuvent détecter de nombreux types de problèmes, allant de la contention sur des verrous [7] aux schémas de communication MPI inefficaces[8]. Cependant, la correction du problème de performance reste à la charge du développeur.

Dans ce projet, nous proposons de combler le fossé entre les outils d'analyse de performance et les systèmes d'exécution afin de réduire la charge du développeur. Pythia vise à fournir aux supports d'exécution des informations extraites de l'analyse de performance. Ces informations pourraient être utilisées par les supports d'exécution pour prendre les décisions les plus judicieuses pour maximiser les performances de l'application.

C.3 Approche scientifique et technique

Le groupe Parallel & Distributed Systems (PDS) de Télécom SudParis mène des recherches sur l'informatique distribuée et les systèmes parallèles. Le groupe a pour objectif d'améliorer les performances, la prévisibilité, la sécurité et la conception des systèmes parallèles et distribués utilisés en HPC ou en Cloud computing.
Le projet Pythia adopte une approche scientifique expérimentale. Nous concevons des systèmes qui sont implémentés sous la forme de prototypes logiciels. Ces prototypes sont évalués au cours d'expérimentations consistant à exécuter des applications parallèles réelles sur des grappes de machines, et en étudiant les performances mesurées.
Les prototypes sont conçus de telle sorte que leur utilisation par un développeur soit la plus simple possible. Il n'est donc pas nécessaire de modifier son application ou de la recompiler. Si cette approche apporte quelques contraintes dans les choix de conception, elle facilite les expérimentations sur des applications réelles.
Par ailleurs, tous les logiciels développés au cours de ce projet sont disponibles en ligne sous licence open-source.

C.4 Résultats obtenus

Les principales contributions du projet sont :
- la conception d'EasyTraceAnalyzer, d'un outil d'analyse de traces générique
- la conception de la librarie Pythia qui fournit un service de collecte de trace et de prédictions pour les supports d'exécution
- le développement de deux supports d'exécution exploitant les prédictions de Pythia

Une première partie du projet a consisté à étudier l'analyse de traces d'exécutions *post-mortem*. Pour cela, nous avons étendu l'outil de collecte de trace EZTrace[1] afin d'en améliorer les performances et de faciliter son utilisation (livrable 1.1). Cela a permis la production de traces de nombreuses applications s'exécutant dans des contextes divers[2] (livrable 4.2). Pour exploiter ces traces, nous avons développé l'outil d'analyse de traces EasyTraceAnalyzer[3] (livrable 1.2). EasyTraceAnalyzer fournit une interface pour développer des analyses de traces. Le cœur d'EasyTraceAnalyzer permet de charger des traces de plusieurs formats (OTF2[9], Pajé[10], LiTL[11], etc.) et il fournit des interfaces pour les manipuler. Cela a permis de développer plusieurs outils capables de fusionner des traces multi-niveaux[12], détecter des problèmes de

[1]Disponible sous licence open-source à cette adresse : https://gitlab.com/eztrace/eztrace
[2]Disponible à cette adresse : https://gitlab.com/eztrace/eztrace-trace-collection
[3]Disponible sous licence open-source à cette adresse : https://gitlab.com/parallel-and-distributed-systems/easytraceanalyzer

contention[12], [13], ou de détecter les principales structures algorithmiques d'un programme[14].

Les enseignements tirés du développement d'EasyTraceAnalyzer nous ont permis par la suite d'effectuer les analyses de traces pendant l'exécution de l'application. Nous avons pour cela développé la bibliothèque Pythia[4] (livrable 1.3) qui a pour but de capturer le comportement d'une application lors d'une première exécution, puis de fournir des prédictions aux supports d'exécution lors des exécution ultérieures. Lors de la première exécution, Pythia collecte des événements (typiquement le début ou la fin de certaines fonctions) et, plutôt que de les stocker sous forme d'une trace, le stocke sous la forme d'une grammaire[14]. La représentation sous la forme d'une grammaire permet de détecter la structure algorithmique du programme, de limiter les ressources mémoire consommées, tout en permettant l'enregistrement d'événements à faible coût. Lors des exécutions ultérieures de l'application, Pythia compare le déroulement de l'application avec la grammaire représentant le comportement général de l'application. Un support d'exécution peut alors demander à Pythia des prédictions sur les prochains événements à venir. Grâce à ces prédictions, le support d'exécution peut prendre des décisions en sachant avec précision le comportement futur de l'application.

Afin d'évaluer la bibliothèque Pythia, nous avons développé deux supports d'exécution qui utilisent l'oracle. Pythia-MPI[5] (livrable 3.1) est un support d'exécution qui intercepte les appels aux fonctions MPI, et qui enregistre des événements avec Pythia. Par ailleurs, Pythia-MPI demande des prédictions et vérifie l'exactitude des prédictions. Ce support d'exécution est une preuve de concept qui nous a permis d'évaluer la précision des prédictions de Pythia sur 13 applications MPI.
Pythia-OMP est un autre support d'exécution exploitant les prédictions de Pythia développé dans le cadre de ce projet[6] (livrable 2.1). Pythia-OMP se base sur le runtime GNU OpenMP qui est en charge de gérer les threads et la répartition des tâches de calcul de programmes OpenMP. Pythia-OMP étend GNU OpenMP en sélectionnant le nombre de threads optimal pour chaque région parallèle. Pour cela, Pythia-OMP exploite les prédictions de Pythia.

C.5 Exploitation des résultats

Le logiciel de collecte de trace EZTrace est utilisé à des fins d'enseignement et de recherche dans plusieurs universités en France ou à l'étranger.

Le PEPR exploratoire NumPex comporte un axe de recherche sur les outils pour le HPC. Dans le cadre du projet NumPex, des travaux seront menés sur l'analyse de performance et devraient s'appuyer sur certains résultats du projet Pythia tels que l'algorithme de détection de la structure d'un programme, la réduction sous forme de grammaire, ou le mécanisme d'oracle.

C.6 Discussion

Les résultats du projet Pythia sont très positifs, et ouvrent la voie à de nouvelles optimisations au sein des supports d'exécution exploitant les capacités d'oracle de Pythia.
Le projet Pythia a montré que de nombreuses applications parallèles ont un comportement prédictible d'une exécution à l'autre, y compris lorsque la taille du problème traité change.

[4]Disponible sous licence open-source à cette adresse: https://github.com/libPythia/pythia
[5]Disponible sous licence open-source à cette adresse: https://github.com/libPythia/pythia_mpi
[6]Disponible sous licence open-source à cette adresse:
https://github.com/libPythia/Pythia_cluster22_reproductibility/tree/main/runtimes/gnu_openmp

Le projet a également montré que la bibliothèque Pythia permet d'effectuer des prédictions de manière précise et à très faible coût.

Dans le cadre du projet, l'utilité de la bibliothèque Pythia a été démontrée en développant deux preuves de concept de supports d'exécution exploitant les prédictions. Le développement dans des runtimes de nouvelles optimisations se basant sur les prédictions de Pythia nous paraît une perspective prometteuse. Le caractère générique de la bibliothèque Pythia devrait permettre son intégration dans n'importe quel type de runtime. Par ailleurs, le faible coût des prédictions (de l'ordre de la microseconde) permet d'envisager des optimisations à grain fin.
Le projet Pythia a également proposé de nouvelles techniques de collecte de trace reposant sur les structures algorithmiques du programme. Ces avancées ouvrent la voie à de nouveaux types d'outils de collecte et d'analyse de traces exploitant le caractère structuré des traces, là où l'état de l'art voit une trace comme une séquence d'événements non structurée. La collecte de grandes quantités d'événements pourrait ainsi se faire avec un faible surcoût et générer des fichiers de traces de petite taille. Une autre piste intéressante consisterait à concevoir un outil de visualisation de trace exploitant le caractère structuré de la trace afin de permettre la visualisation de très grandes traces (plusieurs dizaines de gigaoctets) sur une machine aux capacités modestes telle qu'une ordinateur portable.

C.7 Conclusions

Le projet Pythia a permis de développer un nouveau type d'outils d'analyse de performance capable de guider les supports d'exécution. Les différents logiciels développés dans le cadre de ce projet ont été distribués sous forme de logiciels libres, et ont fait l'objet de publications scientifiques et de présentations.
Les résultats du projet Pythia ouvrent la voie à de nouveaux types d'optimisations dans les supports d'exécution.

C.8 Références

[1] P. Cao, E. W. Felten, A. R. Karlin, et K. Li, « Implementation and performance of integrated application-controlled file caching, prefetching, and disk scheduling », *ACM Trans. Comput. Syst.*, vol. 14, nº 4, p. 311-343, nov. 1996.

[2] X. Ding, S. Jiang, F. Chen, K. Davis, et X. Zhang, « DiskSeen: Exploiting Disk Layout and Access History to Enhance I/O Prefetch », in *USENIX ATC*, p. 14, 2007.

[3] F. Broquedis, N. Furmento, B. Goglin, R. Namyst, et P.-A. Wacrenier, « Dynamic Task and Data Placement over NUMA Architectures: an OpenMP Runtime Perspective », in *IWOMP 2009*, p. 15.

[4] L. Adhianto *et al.*, « HPCTOOLKIT: tools for performance analysis of optimized parallel programs », *Concurrency Computat.: Pract. Exper.*, p. 0-7, 2009.

[5] M. Geimer, F. Wolf, B. J. N. Wylie, et B. Mohr, « Scalable parallel trace-based performance analysis », in *EuroPVM/MPI*, 2006, p. 303-312.

[6] B. Mohr et F. Wolf, « KOJAK–A tool set for automatic performance analysis of parallel programs », in *EuroPar*, 2003, p. 1301-1304.

[7] N. R. Tallent, J. M. Mellor-Crummey, et A. Porterfield, « Analyzing lock contention in multithreaded applications », in *PPoPP*, p. 11, 2010.

[8] M. Geimer, F. Wolf, B. J. N. Wylie, E. Ábrahám, D. Becker, et B. Mohr, « The Scalasca performance toolset architecture », *Concurrency Computat.: Pract. Exper.*, p. 702-719, 2010.

[9] D. Eschweiler, M. Wagner, M. Geimer, A. Knüpfer, W. E. Nagel, et F. Wolf, « Open trace format 2: The next generation of scalable trace formats and support libraries », in *Applications, Tools and Techniques on the Road to Exascale Computing*, 2012, p. 481-490.

[10] L. M. Schnorr, B. de Oliveira Stein, J. C. de Kergommeaux, et G. Mounié, « Pajé trace file format », Version 1.2. 5. Technical report, Laboratoire d'Informatique de Grenoble, France, 2013.
[11] R. Iakymchuk et F. Trahay, « LiTL: Lightweight Trace Library », INF - Département Informatique, Technical Report, juill. 2013.
[12] M. I. Naas *et al.*, « EZIOTracer: unifying kernel and user space I/O tracing for data-intensive applications », *SIGOPS Oper. Syst. Rev.*, vol. 55, n° 1, p. 88-98, juin 2021,
[13] M. S. M. Bouksiaa *et al.*, « Using Differential Execution Analysis to Identify Thread Interference », *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, n° 12, p. 2866-2878, déc. 2019.
[14] A. Colin, F. Trahay, et D. Conan, « PYTHIA: an oracle to guide runtime system decisions », in *Cluster*, 2022.

D LISTE DES LIVRABLES

Date de livraison	N°	Titre	Nature	Partenaires	Commentaires
T+6	1.1	Outil de collecte de trace	Logiciel	Télécom SudParis	
T+15	1.2	Outil d'analyse de trace	Logiciel	Télécom SudParis	
T+18	1.3	Bibliothèque capable de prédire le comportement futur d'une application	Logiciel	Télécom SudParis	
T+24	2.1	Support d'exécution capable d'optimiser le placement des threads.	Logiciel	Télécom SudParis	Remplacé par le livrable 2.2
T+24	2.2	Support d'exécution capable d'adapter le nombre de threads OpenMP à la situation	Logiciel	Télécom SudParis	
T+30	3.1	Support d'exécution capable de précharger les données d'entrées/sorties	Logiciel	Télécom SudParis	Remplacé par le livrable 3.2
T+30	3.2	Support d'exécution capable de prédire les prochaines communications MPI	Logiciel	Télécom SudParis	
T+36	4.1	Evaluation of Tasks 2 and 3 on real application	Rapport	Télécom SudParis	
T+36	4.2	A website with a trace repository		Télécom SudParis	

E IMPACT DU PROJET

E.1 INDICATEURS D'IMPACT

Nombre de publications et de communications

		Publications multipartenaires	Publications monopartenaires
International	**Revues à comité de lecture**	[J1, J2]	
	Ouvrages ou chapitres d'ouvrage		
	Communications (conférence)	[C1, C2, C3]	
France	**Revues à comité de lecture**		
	Ouvrages ou chapitres d'ouvrage		
	Communications (conférence)	[N1, N2]	
Actions de diffusion	**Articles vulgarisation**		
	Conférences vulgarisation		
	Autres		

Autres valorisations scientifiques

	Nombre, années et commentaires (valorisations avérées ou probables)
Brevets internationaux obtenus	
Brevet internationaux en cours d'obtention	
Brevets nationaux obtenus	
Brevet nationaux en cours d'obtention	
Licences d'exploitation (obtention / cession)	
Créations d'entreprises ou essaimage	
Nouveaux projets collaboratifs	
Colloques scientifiques	
Autres (préciser)	

E.2 LISTE DES PUBLICATIONS ET COMMUNICATIONS

Journaux internationaux

[J1] M. Islam Naas *et al.*, « EZIOTracer: Unifying Kernel and User Space I/O Tracing for Data-Intensive Applications », *SIGOPS Oper. Syst. Rev.*, vol. 55, n° 1, p. 88-98, juin 2021, doi: 10.1145/3469379.3469391.

[J2] M. S. M. Bouksiaa *et al.*, « Using Differential Execution Analysis to Identify Thread Interference », *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, n° 12, p. 2866-2878, déc. 2019, doi: 10.1109/TPDS.2019.2927481.

Conférences internationales

[C1] A. Colin, F. Trahay, et D. Conan, « PYTHIA: an oracle to guide runtime system decisions », in *IEEE Cluster*, Heidelberg, Germany, 2022.

[C2] T. T. Nguyen *et al.*, « Why Globally Re-shuffle? Revisiting Data Shuffling in Large Scale Deep Learning », in *IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, Lyon, 2022, p. 1085-1096. doi: 10.1109/IPDPS53621.2022.00109.

[C3] A. Lescouet, E. Brunet, F. Trahay, et G. Thomas, « Transparent Overlapping of Blocking Communication in MPI Applications », in *IEEE 22nd International Conference on High Performance Computing and Communications (HPCC)*, Yanuca Island, Cuvu, Fiji, 2020, p. 744-749. doi: 10.1109/HPCC-SmartCity-DSS50907.2020.00097.

Conférences nationales

[N1] A. Colin, F. Trahay, et D. Conan, « PYTHIA : un oracle pour guider les décisions des runtimes », in *Compas*, Amiens, 2022.

[N2] A. Colin et F. Trahay, « Factorisation de traces d'exécutions de programmes pour l'analyse et la prédiction », in *Compas*, 2020.

E.3 Liste des éléments de valorisation

Logiciels créés dans le cadre du projet Pythia

- **EasyTraceAnalyzer** – Disponible sous licence open-source à cette adresse : https://gitlab.com/parallel-and-distributed-systems/easytraceanalyzer

- **Pythia** – Disponible sous licence open-source à cette adresse: https://github.com/libPythia/pythia

- **Pythia-MPI** – Disponible sous licence open-source à cette adresse: https://github.com/libPythia/pythia_mpi

Logiciels améliorés dans le cadre du projet Pythia

- **EZTrace** – Disponible sous licence open-source à cette adresse : **https://gitlab.com/eztrace/eztrace**

Plates-formes

- **EZTrace trace collection** (dataset) – Disponible à cette adresse : https://gitlab.com/eztrace/eztrace-trace-collection