Pythia: runtime decisions based on prediction (duration: 36 months)

Pythia– Common 2018 ANR call - JCJC (Jeunes chercheuses/Jeunes chercheurs)
Défi 7 – Axe 2: Sciences et technologies logicielles

François Trahay – Télécom SudParis

Contents

Abstract

Runtime systems have to take decision that are critical for the performances of parallel applications. Unfortunately, these decisions can only use heuristics based on the current status of the application in order to estimate how it will behave in the future. As a consequence, runtime systems may take decisions that degrade performances instead of improving them.

Pythia aims at providing runtime systems with means to accurately predict the future. For this, Pythia relies on the deterministic nature of most parallel applications: most programs will behave similarly from one run to another. Thus, we will design a tool-chain that analyzes the execution of a program to provide hints to the runtime systems during future executions of the same program. Thanks to these hints, a runtime system could base its decisions on both the current status of the application, and the future behavior of the program.

Evolutions as compared to the first-round proposal

The current proposition is in line with the pre-proposition on the research directions considered. However, we made some modifications in order to take into account the evaluations. Specifically, we have extended the international and national collaborations as part of the project:

- A collaboration has been initiated with an international partner (University of Chongquing, China). We intend to involve them in Pythia as part of one of the Tasks. As a result, the travel budget has been increased to take into account this collaboration.
- A recent collaboration with national partners from CEA and INSA Lyon has been initiated. We plan to continue this work as part of Pythia. As a result, we integrated this collaboration in one of the planned Tasks.
- François Trahay recently initiated collaboration in industrial partners in France as part of a proposal for a FUI project. Due to his participation, François Trahay will now invest 66 % of this time (24p.m) instead of the 80 % that were previously planned. However, we expect that the results of Pythia could be transferred to industrial partners as part of another FUI project.

Summary of people involved in the project

Partner	Name	Position	Involvement	Role
Télécom SudParis	François Trahay	Associate professor	24 p.m (66 %)	Project leader: he will ensure that everything goes according to the plan. He will co-supervise the PhD student.
Télécom SudParis	Gaël Thomas	Professor	3 p.m	He will help supervising the PhD student.
Télécom SudParis	*To be hired*	PhD student funded by ANR	36 p.m	He/she will work on the oracle library and runtime systems
Télécom SudParis	*To be hired*	Research intern funded by ANR	6 p.m	He/she will design thread scheduling strategies
Télécom SudParis	*To be hired*	Research intern funded by ANR	6 p.m	He/she will help running experiment and contribute to the project valorisation

Context, positioning and objectives

High performance computing has become the cornerstone of many scientific fields ranging from medical research to climatology. Efficiently exploiting a supercomputer is difficult. In addition to expressing the parallelism of an application, it is necessary to manage the hardware resources of the machine. The way an application uses the hardware resources may have a significant impact on the performance: for instance, allocating memory correctly may improve an application execution time by 3.6x relative the default allocation strategy [1].

Runtime systems take the responsibility of exploiting the hardware in place of the program developer by providing an interface to the application. The task of defining efficient strategies for a particular hardware setting thus falls to the runtime, and the developer can focus on algorithmic. When selecting a strategy, runtime systems analyze how the application behaved in the past in order to predict how it will behave in the future. For example, some I/O libraries analyze the data read requests of an application in order to read the next blocks of data in advance and to improve the performance of I/O operations [2, 3].

However, the runtime only reacts to the application instead of anticipating its behavior. This leads to multiple problems as some early decisions may conflict with later ones. For instance, the location where memory is initially allocated has a significant impact on further memory operations and the overhead of moving a memory page closer to a thread is significant for the application performance [4]. Reacting to the application is also a problem for irregular applications where predicting the future behavior based on past event is difficult.

On the over hand, years of research on performance analysis have led to many tools that are able to analyze an application behavior and to identify probable causes of bad performance [5–7]. Because performance analysis tools collect various information during a program execution, and analyze them *post-mortem*, the cost of analysis can be high, whereas a runtime system has to act quickly in order to reduce its impact on the performance of the application. Thus, performance analysis tools can detect many types of problems ranging from lock contention [8], to inefficient MPI communication patterns [9]. However, the task of fixing the performance problem remains the responsibility of the developer.

In this project, we propose to bridge the gap between performance analysis tools and runtime systems in order to reduce the burden on the developer. Pythia aims at providing runtime systems with information extracted by performance analysis that would help runtime system take decisions.

Positioning

Using performance analysis to improve the performance

In order to improve the performance of parallel application, developers need to understand the current performance of their applications and to detect the causes of performance degradation. Performance analysis a been studied for a long time, and many tools for helping developers have been proposed. Many profiling tools [10–13] allow users to spot the hot functions of an application. This information can be used by developer to identify the functions that are worth optimizing. Another class of tools consist in generating trace files depicting an application execution [14–16]. User can then navigate through the trace files using visualization tools [17–19] to manually find inefficiencies.

However, as the trace file grows, it becomes necessary to automate the analysis of traces. Scalasca or Periscope [9, 20, 21] use a database of common inefficient parallel programming patterns that allow users to detect some of the application problems. Other approaches for finding performance bottlenecks consist in comparing the relative performance of functions [22], searching for function whose duration varies [23], or analyzing hardware counters [24]. Based on the output of these performance analysis tools, an application developer can identify the cause of a performance problem, and modify the application to improve its performances.

Performance analysis tools can also give useful hints on which configuration will bring the best performance for an application [25]. TreeMatch analyzes the communication between MPI ranks, and computes the optimal placement of MPI processes on a cluster of machine [26]. In this work, the performance is improved by more than 15 % by selecting the most appropriate configuration setting [27]. A similar approach can be used for binding the threads of an application [28].

Using performance analysis is a good solution for improving the performance of a parallel application. However, application developers need to interpret the output of the analysis tool in order to modify the application algorithms or to adapt the application configuration.

Runtime systems heuristics

Another class of software permits to improve the performance of applications by using heuristics to efficiently exploit the hardware resource. Runtime systems provide applications with an interface (for sending messages to other processes, for synchronizing threads, for running tasks, etc.), and are in charge of exploiting the resources. To do so, runtime systems use heuristics that take decisions (for instance, binding a thread to a particular cpu). These heuristics usually take into account the current status of the application (how many threads are running, how many cpu are available, etc.)

When deciding which execution strategy to apply, runtime systems try to predict the future behavior of an application. StarPU estimates the duration of a task to execute on a computing resource by analyzing the past execution of similar tasks [29].

The affinity between threads and their data has been extensively studied. Most work focus on detecting affinity problems in order to react to them [1,30]. Some other work proactively place data and threads in order to improve the locality of memory access. For this, predicting the affinity between threads is mandatory. This can be predicted for some programming models like OpenMP [31]. Thus, these work either react to bad locality at runtime, or statically place threads and memory at the application startup. None of these work can proactively adapt the thread placement depending on the application phase.

Prefetching disk I/O has been extensively studied for decades. The basic idea of prefetching is to predict the blocks of data that are likely to be read in the future in order to load them in advance and to overlap the disk access with computation [2,3,32]. While analyzing past I/O access permits to predict the future access for regular access patterns, other techniques have to be applied for irregular access patterns. Early works have focused on means for the developer to provide hints on future I/O operations to the I/O library [33]. This allows irregular access patterns to be predicted, but at the cost of a manual modification of the application. Some work try to predict the future access by speculatively executing the application in order to detect future I/O requests [34].

All the existing work either react to the application behavior by analyzing the events that happened during a temporal window, or use heuristics that apply a strategy once and for all. Instead, we propose to proactively take runtime decisions based on predictions of the future behavior of the application.

Position in relation to national/european projects

As presented in Table 1, Pythia relates to several French and European research projects.

Project Name	Type	Description
DASH	ANR 2017-2021	The goal is to analyze the I/O behavior of parallel applications and to generate profiles of these applications. The project also plans to develop new job scheduling algorithms that would use the profiles in order to optimize the job allocation.
MADAME	FP7 2011-2015	This project has the goal of helping programmers develop efficient parallel programs. It provides extensions of the OpenMP profiling and a modeling approach to analyze the performance vs. productivity trade-off.
AUTOTUNE	FP7 2011-2015	The AUTOTUNE (Automatic Online Tuning) has the goal to extend Periscope, an open source automatic online and distributed performance analysis tool developed by Technische Universität München.
Compat Allscale NLAFET	H2020 (2015-2018) H2020 (2015-2018) H2020 (2015-2018)	These projects aim at helping the application developer exploit efficiently exascale supercomputers. This is done by designing generic multiscale algorithms, new parallel programming models or new numerical libraries. As part of these projects, performance monitoring tools will be developed.
NextGenIO	H2020 (2015-2018)	The NextGenIO project aims at improving the IO subsystem in HPC systems using non volatile memory. As part of the project, performance analysis tools will be designed.

Table 1: Related French and European projects

These works all relate to Pythia in that they aims at improving the performance of parallel applications.

As compared to DASH, Pythia is complementary as the two project target different parts of the execution of parallel applications: DASH focuses on the optimization of the job allocation, while Pythia will optimize the use of resources while the application is running.

As compared to MADAME, Pythia is different as it is not bound to OpenMP applications: it targets any HPC application, including distributed applications. Besides, MADAME only focuses on the performance analysis, and it is up to the user to optimize the application.

The AUTOTUNE project is similar to Pythia in that they both target HPC applications in general (ie. running MPI or CUDA). However, AUTOTUNE aims at providing tuning recommendations to the developer, while Pythia aims at providing hints to runtime systems so that they automatically optimize the resource use.

Finally, Pythia also relates to Compat, Allscale, and NLAFET. These projects aim at easing the development of parallel and distributed applications. As compared to these projects, Pythia does not focus on designing new building blocks, but focuses on extracting knowledge from execution traces, and using this knowledge to take decisions at runtime. While some of these projects include the development of new profiling tools, the optimization of an application still falls down to the developer.

Objectives

The main goal of the **Pythia** project is to design a framework that proactively takes decisions using an oracle that provides the runtime with predictions. By knowing the future, the runtime system can take the decisions that will minimize the application execution time.

The general idea of this project is to take advantage of the determinism of most HPC applications to predict their behavior. Most application classes [35] (such as dense linear algebra, or structured grids) apply the same algorithm on the input data. Thus, such an highly deterministic application will have similar behavior from one execution to another. Even more irregular applications (such as sparse linear algebra or unstructured grids) still have regularities: the programs execution flow may take an execution branch or another depending on the input data, but the number of different branches remains limited.

Thus, we propose to design a framework that will capture the application execution, analyze it, and serve as an oracle for the runtime system for future executions of the same application. When running the application again, the runtime dynamically builds a state automaton and compares it to the one provided by the oracle in order to predict the future behavior of the program.

The runtime decisions that can be taken by knowing the future are various and range from HPC-specific optimisations to more general decisions that could improve the performance of web services. Oracle-based decisions could allow runtime systems to schedule threads close to the resources they are about to use, to prefetch data from disks, to reduce the CPU frequency of a task that is not on the critical path, etc.

The Pythia tool-chain will be composed of three key components:

- a trace collection tool for capturing the application execution. The tracing tool intercepts the application calls to a set of functions (MPI functions, OpenMP parallel constructs, etc.) and record events in trace files. This tool will be based on EZTrace [15], a framework for performance analysis developed at Télécom SudParis.
- a post-mortem analysis tool that will analyze the traces, detects recurrent patterns of events, and generate a summary. The output of this tool is a state automaton that describes the successions of interactions between the application and the runtime system. We will extend a pattern discovery tool we developed in the past [36] in order to interact with the other parts of the tool-chain.
- a runtime system that uses the results of the post-mortem analysis tool to decide which strategy is the best-suited. During the execution of the application, a state automaton describing the sequence of events is generated and is compared to the automaton generated offline in order to predict the events that will happen in the near future. The runtime system can thus base its decisions on the events that are likely to happen next.

To demonstrate the applicability of the Pythia tool-chain, we will implement a set of runtime heuristics for various subsystems. While the tool-chain offers many possibilities of optimization, we will focus on two runtime decisions in this project: thread binding based on the used resources, and I/O prefetching.

Thread binding: The affinity between a thread and the resources it uses (network controllers, disks, memory banks, etc.) is critical for performance [37]. As an application runs, its threads use multiple resources that may vary from one phase to another. Thanks to its knowledge of the future behavior of a thread, the runtime system could migrate the thread close to the resource it is about to use extensively in order to improve the locality of the thread and thus the performance.

I/O prefetching Since fetching data from disk is orders of magnitude slower than accessing data from the memory, disk I/O may have a huge impact on the performance of some applications. As a result, predicting which data will be accessed in the future is critical for performance. While regular I/O access patterns can be predicted [2], irregular patterns cannot be predicted from past access only. The sequences of events from the past executions of an application could be used to predict even irregular I/O access that will be issued by a program.

Methodology and risks management

Methodology

The goal of Pythia is to design libraries and runtime systems that improve the performance of parallel applications. Throughout the project, we will constantly run experiments in order to evaluate how the developed software perform.

We will experiment on micro-benchmarks in order to evaluate the overhead of the tool-chain, as well as to assess the performance improvement on applications that show particular patterns.

We will also carry out experiments on real applications running on large machines in order to evaluate how the optimizations transfer to the complexity of real-life programs.

Risk management

The project has two main risks. The first risk is the difficulty to gather execution traces in a portable format. This risk is mitigated by the fact that the tool-chain developed in Pythia will rely on EZTrace, a generic tracing tool developed at Télécom SudParis, that already supports several trace formats (including Pajé and OTF). Thus, extending EZTrace so that it can generate traces in the standard OTF2 format should be straightforward. Moreover, the tool-chain could fall back to other trace formats although this would reduce the compatibility with other performance analysis tools such as Score-P.

Another risk is inherent to any exploratory project such as Pythia. We will define strategies to apply at runtime based on an oracle library. It may turn out that the strategies do not apply to real life applications. To mitigate this risk, we will avoid application-specific strategies when designing the runtime systems. The runtime strategies will also be designed in close collaborate with experts on data placement, and I/O prefetching to mitigate this risk.

Project organisation

Overview

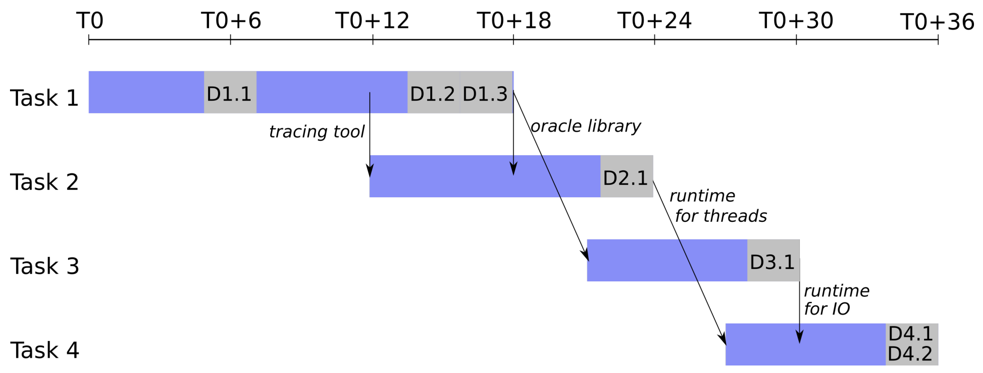

Figure 1: Gantt chart of the project

As described in the Gantt chart in Figure 1, the Pythia project is structured in 4 main tasks:

- **Task 1** will design and implement a tool-chain that captures the behavior of a parallel application, and generate a set of files that describe the captured application.
- **Task 2** will design and implement a runtime system for placing threads on a large machine. The runtime system will use the description files in order to predict which resource the threads are about to use.

- **Task** 3 will design and implement a runtime system for prefetching I/O requests. It will use the description files in order to predict which files and data blocks are about to be used by the application.
- **Task** 4 aims at evaluating the prototypes developed in Tasks 2 and 3 on real applications. It also aims at finalizing the tool-chain so that it can be used in other runtime systems.

Description of the Tasks

Task 1: Extracting state automatons from a trace

leader: François Trahay			
Involved partners	François Trahay	Gaël Thomas	PhD student
Involvement	12	1	15
Input: EZTrace tracing framework, Pattern detection algorithm			
Output: Oracle library			
Start: T_0		**End:** T_0+18	

Deadline	**Delivrable**
T0+6	D1.1: A tracing library able to generate OTF2 trace files (open-source software)
T0+15	D1.2: A trace analysis tool able to detect patterns of events and to generate files that depict the application behavior (open-source software)
T0+18	D1.3: An Oracle library (open-source software)

Goal: To extract useful information from a trace

Detailed work program: There are many performance analysis tools targeting various system components (memory management, network stack, I/O stack, etc.). These tools capture an application execution and provide the application developer with various indicators. It is then up to the developer to use these indicators for improving the application, either by modifying algorithms, or by changing the settings of the application (thread binding, etc.)

The goal of this Task is to develop a tool-chain that will capture the behavior of an application, and generate a set of files describing the application. These files will be used by the runtime systems developed in Tasks 2 and 3 for future executions of the application.

Task1.1: capturing the application behavior

Goal: To run an application and capture its behavior.

Detailed work program: In order to analyze the performance of an application, it is crucial to capture its behavior without altering its execution. The overhead caused by the instrumentation of the application and the collect of traces needs to be as low as possible.

In this Task, we will extend our existing work on the EZTrace tracing tool [15] in order to make it generate trace files in a standard format such as OTF2 [38]. Since EZTrace already generates files in several trace formats (such as Pajé or OTF), we expect that the development cost will be light. However, it will improve the inter-operability of EZTrace will other performance analysis tools.

In this Task, we will also design and implement trace analysis algorithms for detecting the application behavior from a trace file. In a previous work [36], we designed a pattern discovery tool able to detect repetitive sequences of events generated by a thread in a trace. We will extend this work in several ways:

- We will implement a driver so that several trace formats can be read. This will allow the pattern discovery tool to process OTF2 traces generated by either EZTrace or other standard tracing tools (such as the Score-P suite [39]).
- We will extend the pattern matching algorithm so that events generated by multiple threads can be compared. This would permit to differentiate the patterns of events that are common to multiple threads from those that are specific to a thread.
- We will extend the trace analysis tool so that it writes the detected behavior in a set of files that could be used for future execution of the application.

Delivrables:

T0+6 [D1.1]: A tracing library able to generate OTF2 trace files (open-source software)

T0+15 [D1.2]: A trace analysis tool able to detect patterns of events and to generate files that depict the application behavior (open-source software)

Task1.2: designing an oracle library

Goal: To predict the future behavior of a thread.

Detailed work program: In this Task, we will design and implement a library that compares the current execution of a program and its past executions in order to predict the events that are likely to occur in the future. This library will be used in Tasks 2 and 3 for giving hints on the application to runtime systems.

The library we intent to design will base its prediction on the files generated by the trace analyzer developed in Task 1.1. The behavior of an application can viewed as a probabilistic automaton: given a sequence of events, the events that are likely to occur in the future can be predicted based on past execution of the application.

The oracle library will provide an API so that runtime systems can inform the oracle with the events that occur. The API will also provide runtime systems with a way to predict the events that will occur in the future.

Delivrables:

T0+18 [D1.3]: An Oracle library (open-source software)

Risks and backup solutions: The main risk of this task is to not be able to predict future events accurately. We are confident that the risk is low as in recent experiments, we have shown that our existing pattern detection algorithm is able to find repetitive sequences of events in execution traces from many different applications. Because applications are naturally highly structured, an oracle can perform predictions in any case. In a worst case scenario, the predictions could still be performed but a lower accuracy.

Another risk is the incompatibility of the tool-chain with other performance analysis tools such as Score-P. The risk is low as EZTrace already supports several tracing format (including OTF), and we are confident that porting the tool chain to OTF2 can be done. In case the risk materializes, we will fall back to other open trace formats (such as OTF or Pajé).

Task 2: Oracle-based dynamic placement of threads

leader: François Trahay				
Involved partners	François Trahay	Gaël Thomas	PhD student	Intern
Involvement	5	1	8	6
Participants: Kevin Marquet (INSA), Lionel Morel (CEA)				
Input: EZTrace, Oracle library (from Task 1)				
Output: runtime system for thread placement				
Start: T_0+12		**End:** T_0+24		

Deadline	**Delivrable**
T0+24	D2.1: a runtime system that proactively adapts the threads placement (open-source software)

Goal: A runtime system that dynamically places threads

Detailed work program: On large NUMA machines, the thread placement can significantly affect the performance of applications: on some applications, allocating memory correctly may improve the execution time by 3.6x relative to the default allocation strategy [1] ; on high-speed networks, a bad thread placement may degrade the performance by up to 40 % [37]; the performance of disk I/O can vary by up to 33 % depending on the thread placement [40].

In this Task, we will design a runtime system able to proactively adapt the placement of threads. This runtime will provide the oracle library developed in Task 1.2 with information on the threads usage of various resources (network, disk, memory, etc.) The oracle library can then predict the events that are likely to happen in the future. The runtime system can thus use the prediction in order to identify which resources the threads will use in the future, and to adapt the thread binding.

As part of this Task, we will continue a recent collaboration with INSA and CEA that aims at analyzing the memory access patterns of parallel applications [41]. The current work on this topic consists in analyzing the memory access pattern so that the application developer can adapt his program. In Pythia, we intend to automate this process so as to relieve the developer from this task.

Delivrables:

T0+24 [D2.1]: a runtime system that proactively adapts the threads placement (open-source software)

Risks and backup solutions: The main risk of this task is that the designed runtime system cannot improve the performance of a wide range of applications. To mitigate this risk, we will avoid application-specific runtime strategies. In case the designed strategies benefit so only some applications, we will analyze the evaluated applications in order to define the key characteristics that affect the performances of the runtime strategies.

Task 3: prefetching irregular I/O

leader: François Trahay			
Involved partners	François Trahay	Gaël Thomas	PhD student
Involvement	5	1	8
Participants: Jianwei Liao (SWU)			
Input: EZTrace, Oracle library (from Task 1)			
Output: runtime system for I/O prefetching			
Start: T_0+21		**End:** T_0+30	

Deadline	**Delivrable**
T0+30	D3.1: A runtime system that prefetches irregular I/O requests (open-source software)

Goal: A runtime system that predicts future I/O requests

Detailed work program: Due to the huge performance difference between accessing data in memory and accessing data on a disk, optimizing I/O can be crucial for some classes of applications. Prefetching data that will be read in the future allows to overlap I/O operations and computation and can improve the overall performance. Predicting future I/O requests have long been studied [2,33,34], and regular I/O access patterns can be predicted efficiently. However, irregular I/O access patterns are more difficult to predict because I/O predictors only consider the I/O requests previously issued in the current application execution. Relying on both the current execution and past executions would allow I/O prefetchers to predict irregular I/O access patterns.

In this Task, we will design and implement a runtime system able to prefetch I/O data. This runtime system will use the API of the oracle library each time the application performs an I/O request. Based on these events, the oracle will provide the runtime system with prediction of I/O access that are likely to occur in the future. The runtime system may then chose to prefetch the data that will be accessed later.

To pursue this Task, we will work in close collaboration with Jianwei Liao from the Southwest University (SWU) in China. In order to do this, we plan multiple visits, including:

- an early one week visit for François Trahay and the PhD student to SWU. This visit would initiate the collaboration on this task.
- a one week visit for the PhD student to SWU. During this visit, the PhD student would finalize the prototype, and conduct experiments in collaboration with Jianwei Liao.

Delivrables:

T0+30 [D3.1]: A runtime system that prefetches irregular I/O requests (open-source software)

Risks and backup solutions: The main risk of this task is that the proposed prefetching algorithm cannot improve the performance of real life applications. To reduce this risk, we will first study the I/O access patterns of several applications prior to designing the prefetching strategies.

Task 4: Integration and large-scale experimental evaluation

leader: François TRAHAY				
Involved partners	François TRAHAY	Gaël THOMAS	PhD student	Intern
Involvement	2	0	5	6
Input: runtime system for thread placement (from Task 2), runtime system for I/O prefetching (from Task 3)				
Output: Stable version of the Oracle library				
Start: T_0+27		**End:** T_0+36		

Deadline	**Delivrable**
T0+36	D4.1: Evaluation of Tasks 2 and 3 on real application (report)
T0+36	D4.2: A website with a trace repository

Task4.1: Large-scale experimental evaluation

Goal: To test the results of Tasks 2 and 3 on real applications.

Detailed work program: This task aims at evaluating the developed runtime systems and performance analysis tool chain on real life applications running on large machines.

As part of the development of the runtime systems in Tasks 2 and 3, and of the performance analysis tool chain in Task 1, we will evaluate our implementation on small applications and micro benchmark. In this Task, we plan to evaluate our implementation on real applications running on large machines.

This evaluation aims at assessing the scalability of the tool-chain developed in PYTHIA: are the tools capable of analyzing large source codes ? How do they behave when running on many cores ? Can the oracle correctly predict future events on such application ?

Delivrables:

T0+36 [D4.1]: Evaluation of Tasks 2 and 3 on real application (report)

Task4.2: Project valorisation

Goal: To finalize the prototypes so that they can be used by others

Detailed work program: In this Task, we will set up facilities so that the software developed in PYTHIA can be used by the research community. An effort will be put for in stabilizing the source code of the tool chain and to document the tools.

We will also create a trace repository containing the execution traces gathered during PYTHIA. The goal of this repository is to allow other researcher to analyze traces and propose new optimization strategies.

Delivrables:

T0+36 [D4.2]: A website with a trace repository

Risks and backup solutions: There is a risk that we cannot experiment the proposed runtime systems on actual applications. In this case, we will use mini-apps developed for evaluating the I/O subsystem such as the one from the Virtual Institute for I/O [1]. We could also fall back to large-scale mini-apps such as the CORAL benchmark applications [2].

Task schedule, deliverable, and milestones

We sum up the Task schedule graphically in Table 2. A synthetic view of the deliverables is presented in Table 3.

		Year 1				Year 2				Year 3			
		3	6	9	12	15	18	21	24	27	30	33	36
Task 1	Subtask 1.1	🟦	🟦✪	🟦	🟦	🟦✪							
	Subtask 1.2		🟦	🟦	🟦	🟦	🟦✪						
Task 2						🟦	🟦	🟦	🟦✪				
Task 3									🟦	🟦	🟦✪		
Task 4	Subtask 4.1										🟦	🟦	🟦✍
	Subtask 4.2										🟦	🟦	🟦✪
Progress report/Expenses							🟩☆						🟩☆

Legend:
🟦✪ Deliverable is a software 🟦✍ Deliverable is a report 🟩☆ Progress report + expense summary

Table 2: Task schedule for PYTHIA

[1]https://www.vi4io.org/
[2]https://asc.llnl.gov/CORAL-benchmarks/

Task	Title and substance of the deliverable and milestone	Delivery
	T1: Extracting state automatons from a trace	
T1.1: capturing the application behavior		
D1.1	A tracing library able to generate OTF2 trace files (open-source software)	T0+6
D1.2	A trace analysis tool able to detect patterns of events and to generate files that depict the application behavior (open-source software)	T0+15
T1.2: designing an oracle library		
D1.3	An Oracle library (open-source software)	T0+18
	T2: Oracle-based dynamic placement of threads	
D2.1	a runtime system that proactively adapts the threads placement	T0+24
	T3: prefetching irregular I/O	
D3.1	A runtime system that prefetches irregular I/O requests (open-source software)	T0+30
	T4: Integration and large-scale experimental evaluation	
T4.1: Large-scale experimental evaluation		
D4.1	Evaluation of Tasks 2 and 3 on real application (report)	T0+36
T4.2: Project valorisation		
D4.2	A website with a trace repository	T0+36

Table 3: Deliverables and Milestones for Pythia

Requested resources

Cost supported by ANR proposal

Staff Two interns (master-level, 6 months) will be hired for the project (on the 12th month, and on the 30th month) as well as a PhD student (36 months).

The PhD student will work on the design and implementation of the oracle library, as well as on the design of the runtime systems that exploit the oracle predictions.

The first research intern will contribute to Task 2. He/she will work on thread scheduling strategies that take into account the future behavior of threads. The intern will work in close relationship with the PhD student.

The second intern will contribute to Task 4. He/she will help the PhD student running experiments on real life applications. The intern will also contribute to the valorisation of the software developed in Pythia by stabilizing the code, writing documentation, and deploying a data repository where experimental results (such as trace files) will be publicly available.

Mission, traveling, and workshop The traveling costs are evaluated as follows:

- Attending one international conference for one member of the project during 3 years. *Cost: 3 x 3k€= 9 k€*
- Attending one european conference for one member of the project during 3 years. *Cost: 3 x 2k€= 6 k€*
- Three one-week visit to our partner in China as part of Task 2. *Cost: 3 x 3 k€= 9 k€*

Finally, we plan to organize a one-day workshop around performance analysis in conjunction with a major national conference at the end of the 2nd year of the project. The cost (food and breaks) will be partly supported by ANR (20 people): 1000 €.

Hardware & Inward billing Two laptops will be bought for members of the project. *Cost: 2 x 2.5 k€*

In order to run experiments on a large scale machine, we will buy a large NUMA machine able to run 192 threads. According to a quotation, such machine would cost approximately 25 k€.

Cost not supported by ANR proposal

Permanent staff Dr. François Trahay, the principal investor of Pythia, will spend 66 % of his time (24 p.m) working on the project. In addition to supervising the PhD student and the interns, he will participate to the design and development of the software.

Prof. Gaël Thomas will spend 3 p.m working on the project. He will help supervising the PhD student, and will help designing the tools to be developed in the project.

Summary of the project cost

	Cost	Requested funding
Permanent staff / *Frais de personnel permanents*	199 k€	0
Staff / *Frais de personnel*	122 k€	122 k€
Hardware / *Coûts des instruments et du matériel*	30 k€	30 k€
Travel / *Missions*	25 k€	25 k€
Administrative fees / *Frais d'environnement*	14 k€	14 k€
Sub-total / *Sous-total*	**389 k€**	**191 k€**
Requested funding / *Aide demandée*	-	**191 k€**

Consortium around Pythia

Scientific leader

The Principal Investigator of the Pythia project is François Trahay. He will devote sixty-six percent of his research time to the project. François Trahay is an associate professor at Télécom SudParis (TSP) since 2011. He is a member of the HP2 team of the computer science department, which investigates high-performance systems. He received his Ph.D. degree in computer science from the University of Bordeaux in 2009. He has been working on runtime systems for high performance computing since 2006. His research interests now mostly focus on performance analysis for HPC and distributed systems. He is the project leader and main developper of the EZTrace framework for performance analysis. He is the co-author of 19 research papers in top international conferences, workshops and journals such as IEEE TPDS, IPDPS, ICPP, IEEE Cluster, ACM TACO, or IJPP.

Scientific environment

In addition to the principal investigator, the project will involve the HP2 team at Télécom SudParis. HP2 is composed of experts in runtime systems for high-performance computing (Dr. Élisabeth Brunet, and Dr. Amina Guermouche), and for operating systems (Prof. Gaël Thomas). Gaël Thomas will help supervising the PhD student hired for this project.

Pythia will also involve Dr. Jianwei Liao at the Southwest University of China. He is an expert on distributed filesystems. We started working together on prefetching for distributed filesystems in 2015. We will work in close collaboration with Dr. Liao in order to benefit from his expertise in filesystems. As part of Task 2, we plan several trips to the Southwest University of China.

Impact of the proposal

In the short term, we expect that the result of the Pythia project will impact most HPC applications as the problematics of locality and I/O overhead is commonplace. Furthermore, this project addresses several challenges (scheduling, I/O, auto-tuning, etc.) from the Challenges of Exascale Computing [42].

In the long-term, we expect that the Pythia project will change the way runtime systems are designed, and the tool-chain will be used for other runtime decisions. We also expect that the outcome of Pythia could be generalized and applied to other domains. During the project, we will focus on performance improvement, but the collaboration between a performance analysis tool and a runtime system could also be used for reducing the energy consumption without degrading the performance. We also expect that the results of Pythia could be transferred to big-data applications as they have similar properties as HPC applications.

Positionning with respect to the call

This project mainly addresses the "Défi 7 > Axe 2 > Science et Technologies logicielles" challenge. Indeed, Pythia will design mechanisms that allow runtime systems to take better decisions. While the two example runtimes that will be designed will focus on performance of HPC applications, we expect that the mechanisms provided by Pythia could be used to other domains (eg. big data) and to other means (eg. reducing energy consumption). Pythia will also address several *Technical Research Priorities* from the ETP4HPC Strategic Research Agenda [43] including priority #2 System software and management, and priority #3 programming environment.

Valorization strategies, dissemination, and exploitation of results

Software The investigator of the project will continue his commitment to open-source as all the tools developed during the project will be freely available as open-source software. Besides, the software will be maintained beyond the end of the project and efforts will be made to bootstrap a visible and active community of users and contributors, through appropriate channels and organizations.

In addition to software, we also plan to make our research results freely available. We will create a data repository where experiment results such as execution traces, and experiment configuration (eg. a docker file for running the experiment, information on the machine topology, etc.) will be available.

Collaboration We expect that the outcome of the project will trigger collaborations or enforce existing ones. Pythia could be used as part of the new collaborations that we started with INSA Lyon and CEA. Moreover, François Trahay already worked on parallel I/O [32] with researchers for the Southwest University in Chongqing (China). Pythia could be used for continuing such international collaborations.

François Trahay recently initiated a collaboration with DDN, the leading company on I/O architectures for HPC. This collaboration has led us to propose a FUI project that aims at transferring research results (including parts of the EZTrace tracing tool) to industrial partners like DDN, Criteo, or Qarnot computing. We intent to continue this collaboration and to develop new ones. We expect that the results of Pythia (for instance, the runtime system for I/O prefetching) could be part of a supplementary relationship with DDN.

Publication During the project, we intend to publish the scientific results. These results will be published in major international conferences (such as IPDPS, EuroPar, ICPP, ASPLOS, SOSP, etc.) and journals (TPDS, IJPP, etc.) We also intend to disseminate scientific results to the french community as part of the Compas conference.

Reproductibility

The ability to reproduce research results is a growing concern. Pythia will contribute to this effort in several ways:

- All the tools developed as part of this ANR project will be freely available as open-source ;
- We will create a data repository where the experiment results (for instance, the execution traces), as well as the software configuration (eg. docker file for running the experiment) will be available.

We believe that such data and source repositories will help the research community reproduce research results, and propose new performance analysis algorithms or runtime system strategies.

References

[1] M. Dashti, A. Fedorova, J. Funston, F. Gaud, R. Lachaize, B. Lepers, V. Quema, and M. Roth, "Traffic management: a holistic approach to memory placement on numa systems," in *ACM SIGPLAN Notices*, vol. 48, no. 4. ACM, 2013, pp. 381–394.

[2] P. Cao, E. W. Felten, A. R. Karlin, and K. Li, "Implementation and performance of integrated application-controlled file caching, prefetching, and disk scheduling," *ACM Transactions on Computer Systems (TOCS)*, vol. 14, no. 4, pp. 311–343, 1996.

[3] X. Ding, S. Jiang, F. Chen, K. Davis, and X. Zhang, "Diskseen: Exploiting disk layout and access history to enhance i/o prefetch." in *USENIX ATC*, 2007.

[4] F. Broquedis, N. Furmento, B. Goglin, R. Namyst, and P.-A. Wacrenier, "Dynamic task and data placement over numa architectures: an openmp runtime perspective," in *International Workshop on OpenMP*, 2009, pp. 79–92.

[5] L. Adhianto, S. Banerjee, M. Fagan, M. Krentel, G. Marin, J. Mellor-Crummey, and N. R. Tallent, "Hpctoolkit: Tools for performance analysis of optimized parallel programs," *Concurrency and Computation: Practice and Experience*, vol. 22, no. 6, pp. 685–701, 2010.

[6] M. Geimer, F. Wolf, B. J. Wylie, and B. Mohr, "Scalable parallel trace-based performance analysis," in *European Parallel Virtual Machine/Message Passing Interface Users' Group Meeting*, 2006, pp. 303–312.

[7] B. Mohr and F. Wolf, "Kojak–a tool set for automatic performance analysis of parallel programs," in *European Conference on Parallel Processing*, 2003, pp. 1301–1304.

[8] N. R. Tallent, J. M. Mellor-Crummey, and A. Porterfield, "Analyzing lock contention in multithreaded applications," in *ACM Sigplan Notices*, vol. 45, no. 5, 2010, pp. 269–280.

[9] M. Geimer, F. Wolf, B. J. Wylie, E. Ábrahám, D. Becker, and B. Mohr, "The scalasca performance toolset architecture," *Concurrency and Computation: Practice and Experience*, vol. 22, no. 6, pp. 702–719, 2010.

[10] S. L. Graham, P. B. Kessler, and M. K. Mckusick, "Gprof: A call graph execution profiler," in *ACM Sigplan Notices*, vol. 17, no. 6, 1982, pp. 120–126.

[11] A. C. De Melo, "The new linux'perf'tools," in *Slides from Linux Kongress*, vol. 18, 2010.

[12] C. January, J. Byrd, X. Oró, and M. O'Connor, "Allinea map: Adding energy and openmp profiling without increasing overhead," in *Tools for High Performance Computing 2014*. Springer, 2015, pp. 25–35.

[13] R. Rabenseifner, "Automatic profiling of mpi applications with hardware performance counters," in *European Parallel Virtual Machine/Message Passing Interface Users' Group Meeting*. Springer, 1999, pp. 35–42.

[14] M. S. Müller, A. Knüpfer, M. Jurenz, M. Lieber, H. Brunst, H. Mix, and W. E. Nagel, "Developing scalable applications with vampir, vampirserver and vampirtrace." in *PARCO*, vol. 15, 2007, pp. 637–644.

[15] F. Trahay, F. Rue, M. Faverge, Y. Ishikawa, R. Namyst, and J. Dongarra, "EZTrace: a generic framework for performance analysis," in *CCGrid*, 2011.

[16] D. A. Reed, P. C. Roth, R. A. Aydt, K. A. Shields, L. F. Tavera, R. J. Noe, and B. W. Schwartz, "Scalable performance analysis: The pablo performance analysis environment," in *Scalable Parallel Libraries Conference, 1993., Proceedings of the*. IEEE, 1993, pp. 104–113.

[17] O. Zaki, E. Lusk, W. Gropp, and D. Swider, "Toward scalable performance visualization with jumpshot," *The International Journal of High Performance Computing Applications*, vol. 13, no. 3, pp. 277–288, 1999.

[18] W. E. Nagel, A. Arnold, M. Weber, H.-C. Hoppe, and K. Solchenbach, "Vampir: Visualization and analysis of mpi resources," 1996.

[19] K. Coulomb, A. Degomme, M. Faverge, and F. Trahay, "An open-source tool-chain for performance analysis," in *Tools for High Performance Computing 2011*. Springer, 2012, pp. 37–48.

[20] F. Wolf, *Automatic performance analysis on parallel computers with SMP nodes*. NIC, John von Neumann Institute for Computing, 2003.

[21] S. Benedict, V. Petkov, and M. Gerndt, "Periscope: An online-based distributed performance analysis tool," in *Tools for High Performance Computing 2009*. Springer, 2010, pp. 1–16.

[22] C. Curtsinger and E. D. Berger, "C oz: finding code that counts with causal profiling," in *Proceedings of the 25th Symposium on Operating Systems Principles*. ACM, 2015, pp. 184–197.

[23] T. Ohmann, K. Thai, I. Beschastnikh, and Y. Brun, "Mining precise performance-aware behavioral models from existing instrumentation," in *Companion Proceedings of the 36th International Conference on Software Engineering*. ACM, 2014, pp. 484–487.

[24] C. Xu, X. Chen, R. P. Dick, and Z. M. Mao, "Cache contention and application performance prediction for multi-core systems," in *Performance Analysis of Systems & Software (ISPASS), 2010 IEEE International Symposium on*. IEEE, 2010, pp. 76–86.

[25] M. Attariyan, M. Chow, and J. Flinn, "X-ray: Automating root-cause diagnosis of performance anomalies in production software." in *OSDI*, vol. 12, 2012, pp. 307–320.

[26] E. Jeannot and G. Mercier, "Near-optimal placement of mpi processes on hierarchical numa architectures," in *European Conference on Parallel Processing*, 2010, pp. 199–210.

[27] G. Mercier and E. Jeannot, "Improving mpi applications performance on multicore clusters with rank reordering," in *European MPI Users' Group Meeting*. Springer, 2011, pp. 39–49.

[28] M. Castro, L. F. W. Goes, C. P. Ribeiro, M. Cole, M. Cintra, and J.-F. Mehaut, "A machine learning-based approach for thread mapping on transactional memory applications," in *High Performance Computing (HiPC), 2011 18th International Conference on*. IEEE, 2011, pp. 1–10.

[29] C. Augonnet, S. Thibault, and R. Namyst, "Automatic calibration of performance models on heterogeneous multicore architectures." in *Euro-Par Workshops*, 2009, pp. 56–65.

[30] M. Diener, E. H. Cruz, P. O. Navaux, A. Busse, and H.-U. Heiß, "kmaf: Automatic kernel-level management of thread and data affinity," in *PACT*, 2014.

[31] F. Broquedis, O. Aumage, B. Goglin, S. Thibault, P.-A. Wacrenier, and R. Namyst, "Structuring the execution of openmp applications for multicore architectures," in *IPDPS 2010*.

[32] J. Liao, F. Trahay, B. Gerofi, and Y. Ishikawa, "Prefetching on storage servers through mining access patterns on blocks," *IEEE Transactions on Parallel and Distributed Systems*, vol. PP, no. 99, 2015.

[33] R. H. Patterson, G. A. Gibson, E. Ginting, D. Stodolsky, and J. Zelenka, "Informed prefetching and caching," in *SOSP'95*, 1995, pp. 79–95.

[34] F. Chang and G. Gibson, "Automatic i/o hint generation through speculative execution." USENIX, 1999.

[35] K. Asanovic, R. Bodik, B. C. Catanzaro, J. J. Gebis, P. Husbands, K. Keutzer, D. A. Patterson, W. L. Plishker, J. Shalf, S. W. Williams *et al.*, "The landscape of parallel computing research: A view from berkeley," Technical Report UCB/EECS-2006-183, EECS Department, University of California, Berkeley, Tech. Rep., 2006.

[36] F. Trahay, E. Brunet, M. Mosli Bouksiaa, and J. Liao, "Selecting points of interest in traces using patterns of events," in *PDP*, 2015.

[37] S. Moreaud and B. Goglin, "Impact of numa effects on high-speed networking with multi-opteron machines," in *PDCS*, 2007.

[38] D. Eschweiler, M. Wagner, M. Geimer, A. Knüpfer, W. E. Nagel, and F. Wolf, "Open trace format 2: The next generation of scalable trace formats and support libraries." in *PARCO*, vol. 22, 2011, pp. 481–490.

[39] A. Knüpfer, C. Rössel, D. an Mey, S. Biersdorff, K. Diethelm, D. Eschweiler, M. Geimer, M. Gerndt, D. Lorenz, A. Malony *et al.*, "Score-p: A joint performance measurement run-time infrastructure for periscope, scalasca, tau, and vampir," in *Tools for High Performance Computing 2011*, 2012, pp. 79–91.

[40] T. Li, Y. Ren, D. Yu, S. Jin, and T. Robertazzi, "Characterization of input/output bandwidth performance models in numa architecture for data intensive applications," in *Parallel Processing (ICPP), 2013 42nd International Conference on*, 2013, pp. 369–378.

[41] F. Trahay, K. Marquet, L. Morel, and M. Selva, "NumaMMa: NUMA Memory Manager," 2018. [Online]. Available: https://github.com/trahay/numamma-profiler

[42] S. Ashby, P. Beckman, J. Chen, P. Colella, B. Collins, D. Crawford, J. Dongarra, D. Kothe, R. Lusk, P. Messina *et al.*, "The opportunities and challenges of exascale computing," *Summary Report of the Advanced Scientific Computing Advisory Committee (ASCAC) Subcommittee*, pp. 1–77, 2010.

[43] ETP4HPC, "ETP4HPC strategic research agenda 2015 update."