Improving the Performance of Multiprogrammed Parallel Workloads in Origin2000 Systems

Xavier Martorell, Eduard Ayguadé, Jesús Labarta and Nacho Navarro
Departament d’Arquitectura de Computadors (DAC)
Universitat Politècnica de Catalunya (UPC)
c/ Jordi Girona 1-3, Campus Nord, Mòdul D6, 08071, Barcelona, Spain
{xavim, eduard, jesus, nacho}@ac.upc.es

Abstract
In this paper, we present the evaluation of the Nanos parallel execution environment and its comparison with the native SGIMP environment with respect to the execution of multiprogrammed parallel workloads. The Nanos environment is being developed in the NANOS ESPRIT Project. It considers the SGI Origin2000 machine as its main target for improving the parallel execution of applications.

The results show that the Nanos environment improves overall throughput by 80% and reduces the average execution time of the applications to half the time obtained by the standard SGIMP environment.

1. Introduction
Current parallel shared-memory multiprocessors are complex machines, where a large number of architectural aspects should have to be simultaneously addressed in order to achieve high performance. The quick evolution of parallel machines is being followed by the evolution of the parallel execution environments to take advantage of such new architectural characteristics. Both the user and kernel execution levels are the subject of current research works, trying to achieve a good cooperation between them and with the hardware [2][3][5][7][4][16]. These are the trends in which we have been interested during the development of the NANOS LTR ESPRIT Project (E-21907) [11].

In the Nanos Project, we have focussed in the development of a complete parallel environment to support the execution of multiprogrammed workloads. The goal is to achieve high performance using currently available computers. As a result, the Nanos parallel execution environment has been developed, implemented and evaluated in an Origin2000 machine with 64 processors. In this short paper, we present the main characteristics of the cooperation established between the user and kernel execution levels and its behaviour when executing a parallel application workload.

The rest of the paper is structured as follows: Section 2 sketches the Nanos parallel execution environment; Section 3 explains how the cooperation between the user and operating system levels is achieved; Section 4 presents the evaluation of the execution environment, comparing it with the native SGIMP execution environment as provided by SGI in the Origin2000 machines; finally, Section 5 presents the conclusions.

2. The Nanos parallel execution environment
The structure of the Nanos parallel execution environment consists of three basic levels of operation (application/compiler, user-level execution environment and operating system) and their interfaces. There are three different interfaces, corresponding to each one of the levels of operation. The user interface is based on OpenMP directives. The user-level execution environment interface is offered by a user-level threads library specifically designed to support
automatically parallelized applications. The application and the user-level execution environment use an extended operating system interface to establish a dialog with the operating system, in order to ensure an execution of all the applications as smoother as possible.

At the application/compiler level, the NanosCompiler parallelizes applications annotated with OpenMP directives. The compiler is based on Parafrase-2 [13]. It is a source-to-source compiler. It accepts Fortran applications and it generates parallel Fortran with calls to the user-level execution environment.

The user-level threads library (NthLib) supports the user-level execution environment. It is described elsewhere [10][9].

The complete environment supports the execution of evolving and malleable applications [6]. Evolving applications dynamically request the number of processors they need for execution at any moment. Malleable applications are able to run on any number of processors, adapting their execution when the number of processors allocated to them changes. Parallel applications use the operating system interface (Nanos OS) to request processors to the operating system. The user-level execution environment also uses the operating system interface, at safe points, to adapt the execution of the application to the number of processors allocated.

3. User-kernel cooperation

The three levels of operation offer different opportunities for scheduling, that is, mapping the application work to the physical processors. Our proposal works as follows:

• Each application dynamically informs the user-level execution environment and the operating system about its requirements (number of processors) reflecting the actual degree of parallelism that the application wants to exploit at user-level.
• The operating system distributes processors at least at fixed time slices, taking this information into account. It can also redistribute processors using other events, such as changes in processor requests.
• The application is informed about the operating-system allocation decisions and tries to match the parallelism that it generates to the assigned number of processors.
• The user-level execution environment is in charge of ensuring that the parallelism spawned by the application will execute as smoothly as possible, even when the operating system reallocates processors.

In more detail, when it is time for the operating system to reallocate processors, it applies the current scheduling policy, deciding how many processors each application is going to receive in the next time slice. As a result of this decision, some applications are going to loose processors. Then, the operating system asks these applications for processors to be freed, optionally giving them a certain amount of time (the grace time [8]) to answer to the request by releasing the processors. If an application does not answer to the request in time, or when the grace time for that application is zero, the operating system will forcefully claim back processors through preemption, and inform the application. When some work has been preempted, the application always readapts at user-level. This is done at safe points, by yielding the associated physical processor to the preempted process. Ensuring that all preempted virtual processors are stopped at safe points is very important in order to avoid preemption inside critical sections.
Processors moving to another application are allocated to them. The applications receiving processors are informed in such a way that the processors can participate either in the current parallel execution, if there is work available to perform, or as soon as the application checks the number of processors allocated from the operating system, and it spawns further parallelism.

4. Evaluation of the complete Nanos environment

Along the development of the Nanos Project, we have evaluated the results obtained in the execution of several parallel applications in the Nanos parallel execution environment. We compare these results with the execution of the same set of applications in the native SGIMP execution environment based on IRIX 6.5, the f77 compiler version 7.2.1 and the SGI MP Library, as provided by Silicon Graphics in Origin2000 machines.

The workload for evaluation is built using the applications presented in Table 1. The workload consists of the six applications executed in parallel, requesting the indicated number of processors. All applications start at the same time. When one of the applications terminates, another instance of the same application, with the same processor request is launched automatically. All instances start executing with one processor and they request for more processors when spawning the parallelism for the first time. The LTOMCATV releases P-I processors each time it enters a sequential phase.

The resulting workload execution is visualized through the Paraver tool [12], representing time in the x axis and applications in the y axis (see Figure 1, as an example). The names of the applications are displayed on the left-hand side of the figure, along with the number of processors that they are requesting (enclosed in parenthesis). For each application, an horizontal line is displayed. For each instance of an application, a different color is used to fill the horizontal line. Different colors represent, thus, the execution of the different instances of the corresponding application. Throughout each horizontal line, the points where the lines change their color are also marked using flags, allowing the black and white printing of these images.

Two versions of the workload are executed. The first one is the parallel version of the applications running on top of the SGI MP Library and the IRIX operating system as shipped by Silicon Graphics, with the dynamic adaptation to the number of available processors activated, as is by default (the OMP_DYNAMIC environment variable is set to TRUE). And

<table>
<thead>
<tr>
<th>Application</th>
<th>From benchmark</th>
<th>Sequential execution time</th>
<th>Parallel execution time (speedup)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Processors</td>
</tr>
<tr>
<td>BT</td>
<td>NAS [1]</td>
<td>87.5</td>
<td>16</td>
</tr>
<tr>
<td>TOMCATV a</td>
<td>SPEC95FP [15]</td>
<td>67.6</td>
<td>16</td>
</tr>
<tr>
<td>SWIM</td>
<td>SPEC95FP</td>
<td>106</td>
<td>16</td>
</tr>
<tr>
<td>TURB3D</td>
<td>SPEC95FP</td>
<td>288</td>
<td>24</td>
</tr>
<tr>
<td>HYDRO2D</td>
<td>SPEC95FP</td>
<td>154</td>
<td>16</td>
</tr>
<tr>
<td>SU2COR</td>
<td>SPEC95FP</td>
<td>93.5</td>
<td>8</td>
</tr>
</tbody>
</table>

Table 1: Individual evaluation of the applications executed inside the workloads

a. Inside the workload, the LTOMCATV application is used, consisting of ten iterations of the TOMCATV application. In each iteration, this evolving application requests 1 processor to execute its sequential (I/O) part and 16 processors to execute the parallel (computation) part.
the second version is the NNTH version of the applications running on top of NthLib and the CPU Manager, using the user and kernel interfaces.

The objective of this workload is to show which is the behavior of the execution of a workload in which each application is requesting a number of processors in which its execution as an individual application achieves good speedup. The load of the system while running this workload ranges from 81 to 96 processors, on a 64 processor machine, depending on the current phase of the LTOMCATV application. The workload represents a common system load for this kind of machine.

Figure 1 shows the behavior of the workload running in the MP environment, from the point where the second instance of the BT application is launched (left-most flag) till the termination of the eleventh instance of the BT application (right-most flag). During this period of time, the machine is heavy loaded with 81 to 96 processors requested and the scheduling frameworks and synchronization issues in the respective environments are stressed.

Figure 2 shows the execution of the same workload in the Nanos environment, using the same time scale, and running under the cluster policy. This policy assigns processors in clusters of four to improve application placement [10]. Other policies are explained in [14]. It can be observed that the BT and SWIM applications are clearly benefited in the Nanos environment. The LTOMCATV, TURB3D and HYDRO2D show smaller execution times also.

![Figure 1: Workload execution on the SGIMP environment](image)

The good behavior obtained indicates that the Nanos scheduling framework is well designed to achieve high performance when running parallel workloads with a high degree of parallelism, outperforming the results obtained from the SGIMP environment.

Table 2 shows the number of complete instances of each application executed, and the average execution time and standard deviation obtained for the different applications. The number of instances executed for all the applications is greater when the workload runs on the Nanos environment than in the SGIMP environment. The benefits in the execution of the applications are also observed in that the standard deviation of the execution time of each instance is smaller when running in the Nanos environment. From the data presented, we conclude also that the SGIMP environment penalizes the applications which try to use more processors. This is because of synchronization is more difficult to achieve in this environment due to the processor movements. For example, only two instances of the TURB3D are executed in the SGIMP environment, while the Nanos environment executes up to four
complete instances. This application, requesting 24 processors, benefits from the Nanos scheduling framework, where the movements of processors between applications are communicated to the application and the user-level execution environment helps in solving the preemptions.

Figure 2: Workload execution on the Nanos environment (Cluster policy)

<table>
<thead>
<tr>
<th>Application</th>
<th>Instances</th>
<th>Avg. execution time (in secs.)</th>
<th>Std. deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SGIMP</td>
<td>NNTH</td>
<td>SGIMP</td>
</tr>
<tr>
<td>BT</td>
<td>10</td>
<td>18</td>
<td>22</td>
</tr>
<tr>
<td>LTOMCATV</td>
<td>4</td>
<td>5</td>
<td>44</td>
</tr>
<tr>
<td>SWIM</td>
<td>7</td>
<td>23</td>
<td>26</td>
</tr>
<tr>
<td>TURB3D</td>
<td>2</td>
<td>4</td>
<td>102</td>
</tr>
<tr>
<td>HYDRO2D</td>
<td>4</td>
<td>3</td>
<td>46</td>
</tr>
<tr>
<td>SU2COR</td>
<td>5</td>
<td>5</td>
<td>37</td>
</tr>
<tr>
<td>TOTAL</td>
<td>32</td>
<td>58</td>
<td>36.0</td>
</tr>
</tbody>
</table>

Table 2: Results of the evaluation of the workload

To summarize these results, the total throughput (number of completed instances) obtained in the SGIMP environment is 32 applications, while in the Nanos environment is 58 (80% of increment). Also, the average execution time in the SGIMP environment is 36.0 seconds and 19.6 in the Nanos environment.

5. Conclusions

In this paper, we present the evaluation of the Nanos parallel execution environment, comparing it with the SGIMP execution environment on Origin2000 machines.

The results show that the Nanos environment provides a smoother execution of the parallel applications. The Nanos environment shows better behavior in common heterogeneous workloads consisting of several applications, requesting a different number of processors. When the applications are allowed to exploit parallelism, requesting the number of processors which provide good speedup, the throughput is increased by 80% and the average execution time falls to nearly half in the Nanos environment.
6. Acknowledgements

We would like to acknowledge to our partners in the Nanos project for the insightful discussions during the design and implementation phases of the Nanos environment.

This work has been supported by the European Community under the Long Term Research ESPRIT Project E21907 (NANOS) and the Ministry of Education of Spain (CICYT) under contracts TIC97-1445-CE and TIC98-0511.

7. References


