1. Introduction

The workload of each process can hardly be strictly balanced in irregular applications. Such a challenge is a common concern in the following types of applications:

1. Unstructured Grids,
2. Sparse Linear Algebra,
3. Graph Traversal,
4. Backtrack and Branch + Bound,

Unbalanced processes in irregular applications result in longer completion time and wastage of computational resources if each process is allocated a dedicated core.

While it is hard to give a generic solution to balance the workloads in different processes of irregular partitions, virtualization is a practical technique that can be applied to reduce the computational resource wastage with minimal latency on the completion time of each application.

With each process corresponding to a virtual CPU (vCPU), targets are:
1. Increase the resource utilization ratio by deploying multiple vCPUs on the same CPU;
2. Minor delay in the completion of each application.

Two main requirements:
1. Real-Time Performance Guarantees: Without them, a process with a smaller workload may still be running while other processes are already done. This would increase the completion time of the application.
2. Reconfigurability: Different HPC applications are submitted at different times and the completion time also varies. For better viability, the system must be able to handle reconfiguration while still offering real-time performance guarantees.

With each process corresponding to a virtual CPU (vCPU), targets are:
1. Increase the resource utilization ratio by deploying multiple vCPUs on the same CPU;
2. Minor delay in the completion of each application.

With each process corresponding to a virtual CPU (vCPU), targets are:
1. Increase the resource utilization ratio by deploying multiple vCPUs on the same CPU;
2. Minor delay in the completion of each application.

With each process corresponding to a virtual CPU (vCPU), targets are:
1. Increase the resource utilization ratio by deploying multiple vCPUs on the same CPU;
2. Minor delay in the completion of each application.

2. Solution

To offer real-time performance guarantees and reconfigurability simultaneously, we introduce the real-time virtualized platform RRP-Xen.

RRP-Xen is an implementation of the Regularity-based Resource Partitioning (RRP) model on the popular Xen hypervisor. The RRP model is designed based on the cyclic scheduling approach specified by the ARINC 653 standard, which is designed for real-time applications in virtualized environments. The scheduler strictly follows a cyclic schedule generated in the user space to schedule Virtual Machines (VMs). In short, RRP-Xen includes three major modifications compared to the original Xen, as presented in Fig. 3:

1. The RRP-Toolset in the user space is responsible for generating the schedule of vCPUs;
2. The hypervisor will transfer the cyclic schedule into the hypervisor kernel for the scheduler to access;
3. The ARINC 653 scheduler follows the cyclic schedules sent in on the corresponding CPUs to schedule the vCPUs.

Compared to traditional schedulers, RRP-Xen has three major advantages:
1. By following a pre-generated schedule, the scheduling time is very low and thus minimizes the temporal overhead caused by scheduling.
2. With the pre-generated schedule, the resource supply (execution time a vCPU can obtain during a certain period) is highly predictable with real-time performance guarantees proven in previous works with regards to the RRP model.
3. When applications are added into or removed from the system, a new schedule can be generated in the user space and loaded into the hypervisor kernel to reconfigure the scheduler with real-time performance guarantees preserved at all times.

With RRP-Xen, we just need to benchmark the workload of each process on the submitted irregular applications. Then, we configure the vCPUs based on the benchmarked results so that all processes can complete at the same time with the CPU share allocated. The workflow is shown in Fig. 4.

3. Evaluation

Since the support for multiple vCPUs in each VM and the online reconfiguration on RRP-Xen is still under development, we present the benchmarking results of RRP-Xen to validate its real-time performance with each VM (domain) including only one vCPU. Specifically, the metrics we benchmark include:

1. Scheduling Latency: the time spent in the scheduler during 1 second. This metric indicates the temporal cost of a scheduler.
2. Number of Context Switches: the number of context switches between domains during 1 second. This metric reveals the time wasted in saving and loading contexts.
3. Redi-CII scheduling delay: the end-to-end delay for the Redis-server deployed on a certain domain. Table 1 shows that RRP-Xen spends the least time in making scheduling decisions and in saving and loading contexts. This makes the performance of RRP-Xen more predictable with real-time performance guarantees.

Fig. 3: The architecture of RRP-Xen.

Fig. 4: A workflow of the solution based on RRP-Xen.

Table 1: Scheduling Overhead Measurements for a 1 second Trajectory with 2 Idle domains.

<table>
<thead>
<tr>
<th></th>
<th>RRP</th>
<th>RTDS</th>
<th>Credit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scheduling Latency</td>
<td>1.4ms</td>
<td>1.66ms</td>
<td>4.972ms</td>
</tr>
<tr>
<td>Number of Context Switches</td>
<td>67</td>
<td>85</td>
<td>71</td>
</tr>
</tbody>
</table>

4. Conclusion

- RRP-Xen, as the implementation of the RRP model, can provide real-time performance guarantees and reconfigurability needed by a virtualized platform for irregular multi-process applications.
- More automated tools including benchmarking, configuring, and data usage collection need to be implemented for future optimization and applications.

5. References