Aggregating and Managing Memory Capacity Across Computing Nodes in Cloud Environments

Luis A. Garrido[#1], Paul Carpenter[#2]

[#]*Computer Science Department, Barcelona Supercomputing Center*

[1]luis.garrido@bsc.es, [2]paul.carpenter@bsc.es

***Keywords*— memory management, virtualization, tmem**

EXTENDED ABSTRACT

Managing memory capacity in cloud environments is a challenging issue, mainly due to the temporal variability in virtual machine (VM) memory demand. The Virtual Machine Manager or the hypervisor allocates a portion of the physical memory to the VMs, and it can change their allocation dynamically, depending on their needs. In a cloud-coimputing infrastructure, every computing node has an instance of a hypervisor. In many cases, the VMs demand for memory creates too much pressure on the memory resources of a node, prompting the need to make more memory resources available to the computing node. In our research, we have addressed and provided solutions for the two following problems: 1) how to efficiently manage the memory capacity available to a hypervisor? 2) how to aggregate memory capacity across multiple nodes?

A. Memory Management in the Hypervisor

Many solutions exist for dynamic memory management in a single node executing many VMs. Some of these are memory ballooning and memory hotplug, but these are slow to respond and do not provide suitable interfaces for memory aggregation across multiple nodes. Transcendent memory (tmem) was introduced to improve responsiveness in memory provisioning to the VMs by pooling the node's idle and fallow memory in the hypervisor. These pages are given to the VMs on demand through a key-value store.

Still, tmem presents some limitations of its own. State-of-the-art hypervisors do not implement any efficient way to manage the tmem capacity, letting VMs compete for it. Thus, it is possible for VMs to take up a disproportionate amount of tmem capacity, creating an unfair imbalance in the memory allocation, which reduces overall performance.

With our research, we demonstrated the need to implement high-level tmem capacity management, and for this we have designed a mechanism called SmarTmem. This mechanism integrates coarse-grained user-space memory management with fine-grain allocation and enforcement at the virtualization layer.

SmarTmem consists of the following components: 1) hypervisor support, 2) a tmem kernel module (TKM), and 3) user-space process called the Memory Manager (MM). The hypervisor support consists a mechanism to enforce the allocation of tmem pages as dictated by the MM and assign them to the VMs. Additionally, it includes gathering information capabilities regarding the memory utilization of the VMs. The hypervisor sends this information to the MM through the TKM via a virtual interrupt request issued every second, approximately. The Memory Manager keeps track of the status of the nodes over time and reallocates memory dynamically based on its high-level memory management policies. So far, we have implemented the following policies besides the default unmanaged (greedy) way: 1) static allocation, 2) reconfigurable static allocation, 3) and a smart allocation (SM) policy that seeks to reduce the swapping rate proportionally to the sampled demand of the VMs. SM increases the allocation to a VM in increments of P% of the available tmem capacity, and deallocates them by a similar proportion. We tested SM for different values of P.

B. Experimental Platform and Results for SmarTmem

We evaluated SmarTmem using a VirtualBox image with Xen 4.5 and Ubuntu 14.04 with kernel 3.19 in every domain. The VirtualBox environment had two processor cores enabled, 6GB of RAM, 2GB of swap disk and 32GB of disk storage. The physical machine running VirtualBox had an Intel Core i7 processor at 2.1GHz, 8GB of RAM, 4GB of swap disk and 320GB of disk storage. The MM was implemented in C.

We use Cloudsuite benchmarks as our test applications, configuring one VM for each. We run multiple DomUs with different benchmarks, sometimes using the same benchmark multiple times. Table 1 summarizes the scenarios we used. Scenario 1 runs in-memory-analytics twice in every DomU. Scenario 2 runs graph-analytics once in every DomU and Scenario 3 runs graph-analytics in two DomUs and in-memory-analytics in the third one.

TABLE I
EVALUATION SCENARIOS FOR SMARTMEM

Scenarios	VM parameters
Scenario 1	VM1: 1GB RAM, 1 CPU, VM2: 1GB RAM, 1 CPU, VM3: 1GB RAM, 1 CPU
Scenario 2	VM1: 512MB RAM, 1 CPU, VM2: 512MB RAM, 1 CPU, VM3: 512MB RAM 1 CPU
Scenario 3	VM1: 512MB RAM, 1 CPU, VM2: 512MB RAM, 1 CPU, VM3: 1GB RAM, 1 CPU

In Figure 1 we present the running times only for Scenario 3, in which we are able to obtain a peak of 35% improvement over the case with no memory management using SM, and 40% improvement when using static allocation.

C. Aggregating Memory Resources Across Multiple Nodes

In many cases, the memory resources in a node become under pressure due to increase demand of VMs. In light of this, new computer architectures have introduced hardware support for a shared global address space with fast interconnects. These two features are exploited to enable computing nodes to share their resources, resulting in the memory capacity becoming a global rather than a local resource. This helps relieve the pressure on the resources of the node.

In order to aggregate memory capacity across multiple computing nodes, we introduced a mechanism based loosely on our initial approach for memory management within a single node. This mechanism for remote memory aggregation and management we call it AR-Tmem, since it exploits the tmem interface to aggregate memory capacity, an interface very much suitable for these purposes.

AR-Tmem consists of the following three components: 1) hypervisor support for remote memory accesses, page ownership and page transfers, 2) a tmem kernel module for communication between the user-space processes and the

hypervisor, 3) user-space process called the Remote Memory Manager (RMM). The RMMs in every node perform most of the work of AR-Tmem by cooperating to: 1) distribute memory owned by each node among its VMs, 2) distribute memory capacity among nodes, 3) implement the flow of page ownership among nodes, 4) enable nodes to join and leave, and handle failures.

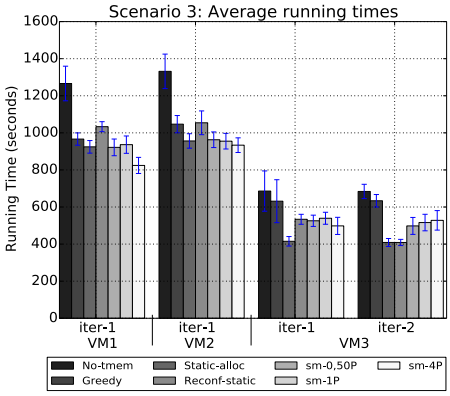


Fig. 1 Average running times for Scenario 3 with different memory management policies.

In our current implementation, we have a centralized approach in which one node has an RMM master (RMM-M) which processes the memory requests of other nodes and forwards them to other nodes when necessary. The RMM-M tracks the status of all nodes by periodically receiving status information from the other RMMs

The hypervisor support includes organization of the memory it owns using a zoned Buddy allocator, with a separate zone for each node in the system (including itself) from which it has ownership of at least one page.

In order to pool the memory resources across multiple nodes, AR-Tmem requires the underlying hardware to have the following: 1) a fast interconnect, providing a synchronous interface across the NUMA architecture, 2) Direct Memory Access from the hypervisors to all the memory available, 3) disable remote access to a node's page on hardware boot, 4) ability to identify the nodes from the physical address of the pages. AR-Tmem implements one high-level policy for memory management aside from the default unmanaged un-aggregated way (greedy-local), which we refer to as Two-Tier Memory Management (TTM). TTM allocates pages by a percentage %P (similar to SM) of the pages owned by the node (local or remote) to a VM depending on its swapping rate, which it attempts to minimize. We refer tas greedy-remote to the case when we aggregate memory, but disable any high-level memory management policies.

D. Experimental Platform and Evaluation for AR-Tmem

We tested our memory aggregation mechanism in a experimental platform consisting of three computing nodes (Nodes 1, 2 and 3). All nodes and VMs run Ubuntu 14.04 with Linux Kernel 3.19+ and Xen 4.5. The RMMs communicate through TCP/IP sockets. We make use of Cloudsuite but with different scenarios. Nodes 1 and 3 execute the scenario under evaluation, while Node 2 executes only the RMM-M. Table II summarizes the scenarios. Scenario 1 runs in-memory-analytics in three DomUs and Scenario 2 executes graph-analytics in three DomUs.

In Figure 2, we show the average running times only for Scenario 1 in Node 1 and Node 3. Greedy-remote obtains a 23.5% maximum improvement over greedy-local, and TTM is able to obtain a 6% maximum improvement over greedy-remote.

TABLE II
EVALUATION SCENARIOS FOR AR-TMEM

Scenarios	VM parameters
Scenario 1	VM1: 768MB RAM, 1 CPU, VM2: 768MB RAM, 1 CPU, VM3: 1GB RAM, 1 CPU
Scenario 2	VM1: 512MB RAM, 1 CPU, VM2: 512MB RAM, 1 CPU

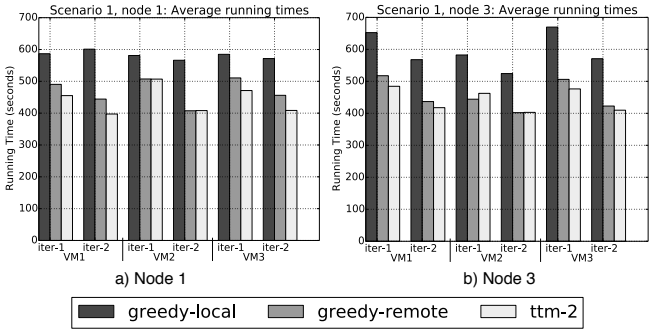


Fig. 2 Average running times for Scenario 1 with different memory management policies. Ttm-2 is the case where P=2.

E. Current Research Efforts

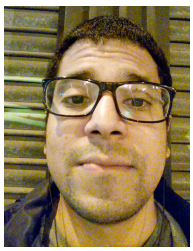
We are developing sophisticated memory management policies that would work well within each node and globally. Resiliency and reliability are one of our main concerns. In addition, we are also studying how our mechanism for memory management and aggregation relates to other dynamic memory management mechanisms.

References

[1] D. Magenheimer, C. Mason, D. McCracken and K. Hackel, "Transcendent Memory and Linux", in Proeedings of the Linux Symposium, pp. 191-200, Citeseer, 2009

[2] Y. Durand, P.M. Carpenter, S. Adami, A. Bilas, D. Dutoit, A. Farcy, G. Gaydadjiev, J. Goodacre, M. Katevenis, M. Marazakis, E. Matus, I. Mavroidis and J. Thomson, "Euroserver: Energy Efficient node for European Micro-servers," in Digital System Design (DSD), 2014 17th Euromicro Conference on, pp.206-2013, IEEE, 2014.

Author biography



Luis Angel Garrido was born in Panama City, Panama. He received the B.E. degree in Electronics and Telecommunications engineering from the Technological University of Panama, Panama, in 2011, and the Master of Science degree in Electrical Engineering and Computer Science from National Chiao Tung University (NCTU), Taiwan, in 2013. Since July 2014, he has been part of the research team at the Barcelona Supercomputing Center as PhD student. His current research interests include computer architecture, memory management, virtualization technology and cloud computing.