Objective of the study

The objective of this study is to examine the use of distributed shared memory vs. local memory in Linux for Matlab matrix operations. Main interest is to discover emergent matrix memory access times for both cases, characterize the caching behavior under Matlab and the extrapolate the applicability to numeric computing of the looked-at ScaleMP distributed shared memory facility.

Distributed Shared Memory

Distributed shared memory (DSM) creates a unified virtual memory region over distributed nodes. Being a special case of non-uniform memory access (NUMA), DSM entails memory aggregation from network-attached remote CPUs vs. that of symmetrical multiprocessor systems linked with processor busses. By coupling memory, DSM can expose a single-system image (SSI) view of the address space. DSM is not a new approach in that in has been applied in symmetric architecture dating to the IBM S/390 class mainframe Parallel Sysplex coupling facility. Various types of DSMs exist with commonality in that DSM usually entails cache-coherence. The NUMA variant cache only memory architecture (COMA) is receiving special attention recently in that by utilizing local nodes for caching of global memory pages, the applicability of the DSM approach is extended to use cases which require (adaptive) local working-set based data availability.

Related DSM/SSI Approaches

The DSM examined in this study is virtual-machine based. Here, the (shared memory) global address space is accessible from within virtual machine (VM). DSMs of this type can be fully-or paravirtualized. A different approach is taken by non-virtualized Kernel memory-management (modulation-based) DSMs such as Kerrighed [3] and LinuxPMI. Transcendental memory [5] also relates to DSM, yet proposes a new distinct layer in the memory hierarchy. Within user-space many libraries provide services akin to DSM, such as for distributed matrices. Adjacent to DSM – within the partitioned global address space (PGAS) paradigm – the library Global Array Toolkit[2] distinguishes between explicitly local addresses space and global memory. Generally, non-message passing proposals such as the X10 language [4] also relates to DSM, yet proposes a new distinct layer in the memory hierarchy.

ScaleMP

ScaleMP is a commercial COMA and virtualization type DSM that besides memory aggragation provides CPU pooling (not in scope for this study). A head-end exposes infiniband network shared memory. ScaleMP cache locality is dynamic with initially (remotely) accessed memory becoming local cache lines - which may get adaptively overwritten.

Test Methods and Process

A comparative timing of (a) spawning matrices and (b) matrix-matrix multiplication in Matlab was conducted on 2 ScaleMP-coupled 32GB nodes vs. a single 64GB node. Exploitative study designs also utilized matrix-vector multiplication and explicit placement of memory for the DSM-nodes on local/remote hosts. As ScaleMP exposes memory LinuX-NUMA compliant, memory-CPU placement experiments were undertaken using numactl(1). While measured memory bandwidth showed to be consistent with expectations for local/remote placement, memory-parameters-only changes gave insufficient granularity for a parameter optimization which would have been of interest in addition to the study scope listed previously. In a further attempt to explicitly place matrix memory, a shared-memory approach was attempted using the Matlab package sharedmatrix[3]. Here, memory on the DSM-node was partitioned in 2 halves, each with a numactl(1) placed Matlab process image.

Environment

The test environment consisted of 3 nodes, each 8 core Intel E5-2680 (Sandy Bridge) CPUs with with hyperthreading, 4 memory channels, and a maximum memory bandwidth of 51.2GB/s. Two nodes with 32GB each were clusters using ScaleMP, a third node with 64MB was used for control-enabling comparison of runs to the local memory case. Interconnect between the DSM nodes was 4xFDR Infiniband at 54.54 Gbit/s effective bandwidth with directly linked (non switched) nodes.

Discussion and Extrapolated Application Scenarios

DSM using ScaleMP showed to be applicable for at least large matrix multiplication. While remote memory bandwidth precludes DSM use in many applications requiring access to the top memory hierarchy tiers, the adaptive COMA caching in the test environment suggests applicability to problems where a transient working set as part of a larger problem data space fits local memory. For example, N-body problems may fit despite de-facto communication constraints. Select big data analytics problems, which without sufficient memory have to resort to streaming, optimizations or e.g. the map-reduce pattern may benefit from COMA DSM as well.

References

[7] ScaleMP, vSMP-5.5.15.27 Fort Lee, NJ USA: ScaleMP, Inc., 2014