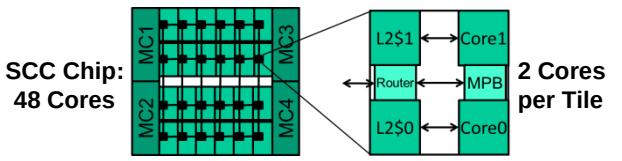
Performance Tuning of SCC-MPICH by means of the Proposed MPI-3.0 Tool Interface

Carsten Clauss, Stefan Lankes, and Thomas Bemmerl

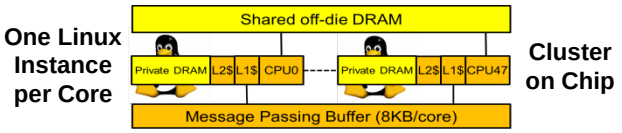
1. The Intel SCC Many-core Processor

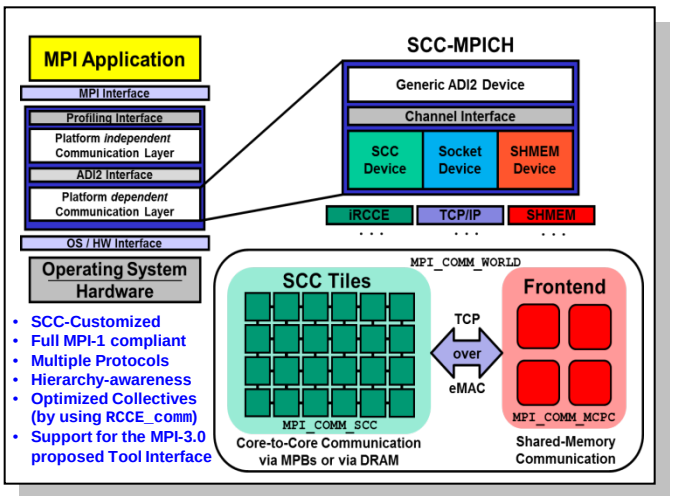
The Single-Chip Cloud Computer (SCC) experimental processor is a 48-core *concept vehicle* created by Intel Labs as a platform for many-core software research. The cores of the SCC are arranged in a 6x4 on-die mesh with two cores per network node. By means of this network, all cores can access a global shared-memory space of up to 64GByte via four on-die memory controllers. In addition to this global off-die shared memory, each core provides a chunk of 8kByte fast on-die memory that is also accessible to all other cores. These additional on-die shared-memory chunks are intended to pass messages directly between the cores, and this is why they are referred to as *Message-Passing Buffers* (MPBs).
In contrast to common multi-core processors, the SCC does not provide any cache-coherency between the cores. For that reason, the global shared-memory is logically distributed in such a way that each core can boot its own Linux image.
Therefore, the architecture of the SCC can be regarded as a *Cluster on the Chip* where message-passing is the programming paradigm of choice.
Intel provides a customized message-passing library for the SCC, called *RCCE* (pronounced "*rocky*"), that utilizes the fast on-die MPBs. In doing so, RCCE offers an application programming interface (API) with a semantics that is derived from the MPI standard. However, while the MPI standard offers a very broad range of functions, the RCCE API is consciously kept small and far from implementing all the features of the MPI standard. For this reason, we have implemented an SCC-customized MPI library, called SCC-MPICH, which in turn is based upon iRCCE, a non-blocking communication extension to the RCCE communication library.

- 48 Pentium-I Cores arranged in a 6x4 on-die Mesh (2 Cores per Tile)
- On-die Message-Passing Buffers (MBP) / 16kB per Tile
- 4 on-die Memory Controllers (MC1-4) / max. 64GByte DDR3 off-die memory



- Strictly *No* Cache Coherency (*Cluster-on-Chip* Architecture)
- *Private* off-die DRAM Regions: Caches *enabled* (One Linux instance per Core)
- *Shared* off-die DRAM Region: Caches *disabled* per default





- SCC-Customized
- Full MPI-1 compliant
- Multiple Protocols
- Hierarchy-awareness
- Optimized Collectives (by using RCCE_comm)
- Support for the MPI-3.0 proposed Tool Interface

2. SCC-MPICH

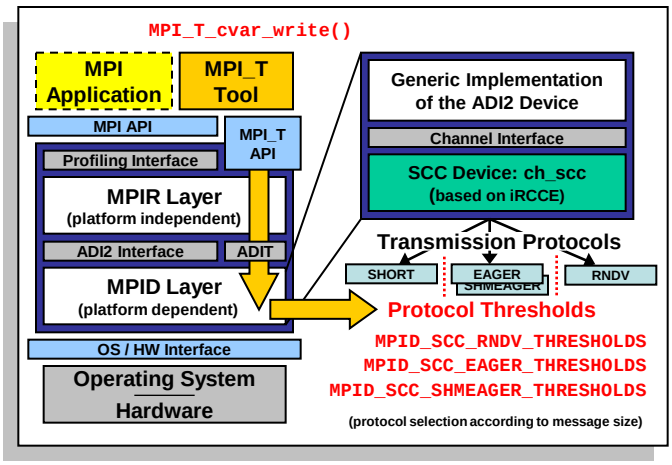
SCC-MPICH is based on MP-MPICH (a multi-platform MPI library that in turn is derived from the original MPICH) as well as on iRCCE as the low-level communication layer. In doing so, we have extended MP-MPICH by a new communication device (ch_scc) that utilizes the fast on-die MPBs as well as the off-die shared-memory for the core-to-core communication. In turn, this new SCC-related communication device provides four different communication protocols: *Short*, *Eager*, *Rendezvous* and *SHM-Eager*.
The Short protocol is low latency optimized and used for exchanging message headers as well as header-embedded short payload messages via the MPBs. Bigger messages must be sent either via one of the two Eager protocols or via the Rendezvous protocol. The main difference between Eager and Rendezvous mode is that Eager messages must be accepted on the receiver side even if the corresponding receive requests are not yet posted by the application. Therefore, a message sent via Eager mode can implicate an additional overhead by copying the message temporarily into an intermediate buffer.
However, when using the SHM-Eager protocol, the off-die shared-memory is used to pass the messages between the cores. That means that this protocol does not require the receiver to copy unexpected messages into additional *private* intermediate buffers unless there is no longer enough *shared* off-die memory.
The decision which of these protocols is to be used depends on the message length as well as on the ratio of expected to unexpected messages.

3. The Tool Information Interface

Currently, the working groups of the MPI-Forum are fostering the development of the upcoming MPI 3.0 standard. In doing so, the so-called Tools Working Group deals with the definition of additional information interfaces that should help to enhance the interaction between MPI implementations and additional tools like debuggers, profilers and performance tuners.
Although the draft for these new interfaces is still under active development, we have already prototyped a major part of the proposed functions on top of MP-MPICH for a use case evaluation. Since SCC-MPICH is based on MP-MPICH, we can already use the new information interface to query and to tune performance and configuration parameters of the SCC-related communication device. That way, we are able, for example, to determine optimal threshold values between the above mentioned communication protocols.

Acknowledgments:

We are not part of the MPI 3.0 Tools Working Group, but we are observing the respective standardization process with great interest.
The development of SCC-MPICH was supported by Intel Corporation. We would like to thank especially Ulrich Hoffmann, Michael Konow and Michael Riepen of Intel Braunschweig for their help and guidance.



LEHRSTUHL FÜR BETRIEBSSYSTEME
Univ.-Prof. Dr. habil. Thomas Bemmerl
RWTH AACHEN UNIVERSITY