Embedding Fault-Tolerance with Dual-Level Agents in Many-Core Systems

Liang Guang · Syed M. A. H. Jafri · Bo Yang · Juha Plosila · Hannu Tenhunen

Abstract Dual-level fault-tolerance is presented on many-core systems, provided by the software-based system agent and hardware-based local agents. The system agent performs fault-triggered energy-aware remapping with bandwidth constraints, addressing coarse-grained processor failures. The local agents achieve fine-grained link-level fault tolerance against transient and permanent errors. The paper concisely presents the architecture, dual-level fault-tolerant techniques and experiment results.

1 Introduction

Due to the increasing influence of PVT variations and complex application scenarios, many-core systems rely on run-time optimization to achieve the expected performance, energy-efficiency and dependability. In particular, as errors and failures may occur unpredictably at various architectural levels, fault-tolerance needs to be provided adaptively both in coarse and fine granularity. Ad-hoc manners of fault management may lead to non-compatible system architectures which are not scalable nor portable. Instead, we propose a generic system architecture with dual-level agents, supporting fault tolerance in many-core systems with diverse granularities.

2 Dual-Level Agents for Run-Time Management Services

A design layer for monitoring and reconfiguration services is added upon the conventional many-core architectures (Fig. 1). The layer is composed of one
system agent, which is the general manager of all monitoring and reconfiguration operations, and distributed local agents, which are delegates of the system agent to actuate the operations. Such organizational partition of agents is designed to support both coarse and fine-grained services, including fault-tolerance. The system agent monitors the global system status, performance and errors, and handles resource allocation. The local agent only monitors the fine-grained status of individual component, such as a router. It reconfigures the local setting, e.g. the replacement of broken wires. As monitors themselves also suffer from errors, the system agent is monitoring the error status of the local agents. The system agent is designed with higher reliability, which will be addressed in the future work.

![Diagram of system and local agents](image)

**Fig. 1** System and Local Agents for Scalable Monitoring and Reconfiguration [Asad et al(2012)]

### 3 Coarse and Fine-Grained Fault-Tolerance with Energy Awareness

We consider a platform where multiple applications can be dynamically loaded, which is a very likely scenario for the thousand-core SoC used as the processing engine in cloud computing. As illustrated in Fig. 2, the system agent, running software instructions, determines the mapping of applications on available processors. To account for fault-tolerance, a portion of processors are left as spares. In case one processor fails, the most suitable spare is utilized by the software agent to replace the broken one. The suitability is chosen, in this case, based on the minimal potential energy consumption of inter-processor communication, given the bandwidth constraint is met (Section 4). While the system agent addresses the processor failure, the local agent, which is a hardware module attached to each router, addresses the link errors. Each local agent detects transient link errors with Hamming codes. If the error persists after repetitive checkings, a permanent fault is identified on a certain wire. The local agent then replaces the broken wire with a spare wire.
4 Initial Experiments

For experimental purposes, four applications (one for image processing, one MPEG encoding, two parallel kernels) are mapped on a 10\(^8\)9 many-core system. The initial mapping is done with multi-application multi-step mapping algorithms [Yang et al.(2010)Yang, Guang, Canhao, Yin, Säntti, and Plosila]. 81 processor cores are used for the initial application mapping (26, 21, 18 and 16 respectively), leaving 9 spare cores. With random choice of processor errors, the fault-triggered remapping searches for the most energy-efficient replacement. Considering that the number of spares is limited, an exhaustive searching is considered a proper tradeoff between the solution’s optimality and run-time overhead. The energy consumption is represented by the weighted communication volume (WCA; mathematical modeling as Volume × HopCounts), which is proportional to the dynamic energy consumption. As the maximal channel bandwidth can not be violated after remapping, the bandwidth constraint is checked for every possible replacement. Table 1 lists four cases of replacements. WCA is reported as normalized to the WCA of the initial mapping. As shown in Table 1, the energy consumption is increased after remapping, as a processor needs to be replaced by a spare more distant from other utilized processors. When the bandwidth constraint is considered, the remapping may have to utilize a spare even further in distance to avoid communication congestion, thus with higher WCA.

The local agent integrates a state machine to automate the link-level fault management (Fig. 3). If the Hamming decoder identifies transient bit errors, the local agent will repetitively check the link. If a bit error persists, the corresponding wire is considered permanently broken. Then a spare wire is used for the replacement. Based on the availability of spares, a number of
Table 1 Exemplified Results of Fault-Triggered Energy-Aware Remapping

<table>
<thead>
<tr>
<th>Failed Nodes</th>
<th>Replacement WCA</th>
<th>Normalized WCA</th>
<th>Bandwidth Constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td>(23,3,22)</td>
<td>(35,5,25)</td>
<td>1.48</td>
<td>No</td>
</tr>
<tr>
<td>(23,3,22)</td>
<td>(35,5,45)</td>
<td>1.64</td>
<td>Yes</td>
</tr>
<tr>
<td>(72,3,22)</td>
<td>(75,5,25)</td>
<td>1.48</td>
<td>No</td>
</tr>
<tr>
<td>(72,3,22)</td>
<td>(75,5,35)</td>
<td>1.56</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Wire errors can be tolerated. If all spare wires are used but more errors occur, a request is sent to the system agent for configuring rerouting or remapping (details beyond the scope of this paper). The local agent is coded in VHDL and synthesized with 65nm CMOS library. Each router (with 136b unidirectional links) is 33806um². Each packet has 128bits, in addition to 8 bits as spare wires. Each local agent (including Hamming coder, decoder and control state machine) requires 9098um² (26.9% area overhead).

Fig. 3 State Transition of Local Agents for Link-Level Fault-Tolerance

5 Conclusion and Future Work

The dual-level agent-based management architecture is modular and scalable. The system agent is exclusively dealing with major failures influencing global performance. The fine-grained local fault detection and recovery are handled by the distributed local agents. The architecture has been experimented with expected performance, with moderate hardware overhead. It is an on-going research, with future work focusing on the completion of fault detection and recovering techniques on each agent level.

References
