Future Generation Computer Systems 53 (2015) 90–99

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

A hierarchical watchdog mechanism for systemic fault awareness on distributed systems

Roberto Ammendola [a], Andrea Biagioni [b], Ottorino Frezza [b], Francesca Lo Cicero [b], Alessandro Lonardo [b], Pier Stanislao Paolucci [b], Davide Rossetti [b,1], Francesco Simula [b], Laura Tosoratto [b,*], Piero Vicini [b]

[a] INFN Sezione di Roma Tor Vergata, Via della Ricerca Scientifica, 1 -00133 Roma, Italy
[b] INFN Sezione di Roma, P.le Aldo Moro, 2 -00185 Roma, Italy

HIGHLIGHTS

- We approach fault tolerance for distributed systems from fault detection and awareness.
- We propose a HW/SW mechanism based on a mutual watchdog mechanism between Host and NIC.
- A double diagnostic message path leads to resilient systemic fault awareness.
- Our tool can interface fault reaction/recovery systems to trigger them automatically.
- Our mechanism has no impact on system performance.

ARTICLE INFO

Article history:
Received 31 July 2013
Received in revised form
7 October 2014
Accepted 28 December 2014
Available online 5 January 2015

Keywords:
Distributed architectures
System-level fault tolerance
Dependable and fault-tolerant systems and networks

ABSTRACT

Systemic fault tolerance is usually pursued with a number of strategies, like redundancy and checkpoint/restart; any of them needs to be triggered by safe and fast fault detection. We devised a hardware/software approach to fault detection that enables a system-level Fault Awareness by implementing a hierarchical Mutual Watchdog. It relies on an improved high performance Network Interface Card (NIC), implementing an *n*-dimensional mesh topology and a Service Network. The hierarchical watchdog mechanism is able to quickly detect faults on each node, as the Host and the high performance NIC guard each other while every node monitors its own first neighbours in the mesh. Duplicated and distributed Supervisor Nodes receive communication by means of diagnostic messages routed through either the Service Network or the *N*-dimensional Network, then assemble a global picture of the system status. In this way our approach allows achieving a Fault Awareness with no-single-point-of-failure. We describe an implementation of this hardware/software co-design for our high performance 3D torus NIC, with a focus on how routed diagnostic messages do not affect the system performances.

1. Introduction

Even if always strictly nonzero, chances of faults in a component (be it a processor, a memory, a storage unit or a network link) of an HPC computing system leading to loss or corruption of work used to be so small that the implementation effort of safety strategies like an explicit checkpointing schedule – to recover from unexpected halts – or duplicated execution – to rule out corruption by double checking the results – could be deemed worthy only in extreme cases.

For petascale (and more so for projected exascale) installations, the situation can change dramatically; for systems assembled with current technology nodes – *i.e.* off-the-shelf components that equip the nodes have a Mean Time Between Failures (MTBF) realistically valued at tens of years – and no other particular provision, the continuous uptime of the whole system can be as low as a few hours. This hints to the fact that, for larger systems, faulting components become an inevitable architectural constraint that implies different strategies and related tradeoffs [1].

For example, redundancy is a well-established approach to fault tolerance; hardware or software elements are replicated – majority voting techniques veto out failing replicas – so that a fault does not immediately tear down a system while waiting to be serviced.

Another widespread practice is rollback/recovery; periodic committal to storage of application snapshots with restart and

* Corresponding author.
E-mail address: laura.tosoratto@roma1.infn.it (L. Tosoratto).
[1] Present address: NVIDIA Corp., Santa Clara, CA, United States.

http://dx.doi.org/10.1016/j.future.2014.12.015
0167-739X/

eventual rework whenever a fault occurs is a standard fault tolerance *reactive* strategy that can be employed at different system levels so as to make it more or less automated, fast and user transparent.

An example of a more sophisticated, *pro-active* strategy is task migration; when equipped with adequate predictor heuristics able to parse available sensor logs, the system has chances of preemptively seeing a fledgling unhealthy status and acting to evade a likely fault, *e.g.* by snapshotting a task running on one core and moving it onto another. Clearly, all these techniques have drawbacks. For example, the frequent and globally consistent checkpoints needed for efficient rollback/recovery put a large strain on storage and network; this can turn them into severe bottlenecks to global performance and dedicated but expensive solutions – fast local storage like phase change SSDs – can only partially mitigate the problem. Because of this, a checkpointing-equipped application has to cope with ever diminishing useful work once run on large number of nodes. An example is given in [2]; under the optimistic assumption of perfect weak scaling, only 35% of the actual runtime of a 168-h (one week) job on a 100k nodes machine accounts for useful work—the remainder is wasted in checkpointing, restarting and recomputation of work lost since last checkpoint.

On the other hand, effectiveness of any proactive technique relies on accurate fault prediction; this represents a complex and very active research area of and by itself, encompassing fault modelling, data mining and statistical signal analysis [3].

Above all, most of the techniques like redundancy and task migration depend on a certain degree of system overprovisioning that can become a cost which is unacceptable for all but mission-critical environments.

Our investigations on fault tolerance were spurred in the context of the EURETILE project [4], aimed at designing and implementing a massively parallel, tiled and fault tolerant computer architecture. In EURETILE we developed, the APEnet+ board—an FPGA-based, high speed network interconnect designed for GPU-accelerated clusters with 3D toroidal topology. Looking at the many issues at hand, it seemed fitting a *divide et impera* approach to the problem, identifying a fault tolerant system as having these two properties:

- it is able to reliably acknowledge the occurrence of a faulty situation, whether it has already befallen or is yet to happen—system has *fault awareness*
- on such basis, it is able to determine and eventually initiate appropriate measures to avoid, overcome or contain the fault—system performs *fault reaction*

and deciding to focus as a first step solely on fault awareness. For this reason, we added logic to APEnet+ supporting a mutual watchdog mechanism between the FPGA and the host processor; we call this logic LO|FA|MO (LOcal FAult MOnitor).

It is common practice for a cluster node to assess the health of the components it hosts (in this case, its network interface) by some interface with onboard sensors; for that, APEnet+ provides stats – temperature, voltage, error levels on its links – that are available to a predicting heuristic in its evaluation for proactive measures against network faults. Not so common is the fact that, with LO|FA|MO, an APEnet+ board can do the reverse: it can autonomously sense when its host misses an update to the shared watchdog – *e.g.* host crashed for some reason – and, thanks to the high connectivity of a 3D toroidal mesh, it can promptly notify the problem to its network neighbours that can relay the information to any supervising entity, be it centralized or distributed. Furthermore, even in case of either the board and the host faulting at the same time – *e.g.* for a local power shortage – boards on first neighbouring nodes can sense their peers going missing and notify the supervisor on their behalf.

In this way, LO|FA|MO is a foundational, platform-agnostic component for fast and reliable fault awareness, on top of which a supervisor can be built.

In Section 2 there is a recap of fault tolerance literature specifically regarding HPC and watchdog systems. An overview of the LO|FA|MO architecture is in Section 3, containing a sketch of the reference platform (this includes the LO|FA|MO implementation host – the APEnet+ board – and a reference platform deployment – the QUonG cluster –), details of the watchdog mechanism and a list of variables that LO|FA|MO can currently query for fault diagnosis. Actual LO|FA|MO hardware implementation is outlined in Section 4 while the software daemon that interfaces LO|FA|MO with the host OS – the Host Fault Manager (HFM) – is described in Section 5. Distributed fault awareness for the system is derived from how LO|FA|MO is designed to exploit the high connectivity of a 3D toroidal mesh; this aspect is discussed in Section 6. In Section 7 it is assessed how quick the current LO|FA|MO implementation is in detecting and diffusing faults information. Future investigations in fault tolerance that we want to pursue with LO|FA|MO are outlined in Section 8; conclusions are in Section 9.

2. Related work

The literature shows a number of monitoring and fault detection tools for HPC clusters. A well known example is Ganglia [5], a scalable monitoring tool for clusters and Grids.

Periodic device polling in order to monitor their liveness, *i.e.* the use of watchdog components, either in hardware or software, is well documented in fault tolerance; this has been applied pervasively to detect faults on distributed systems [6–8]. More widely used at a software level is the similar concept of the *heartbeats* mechanism [9], where devices to be monitored emit a sort of ⟨`I'm alive`⟩ message [10]; obvious drawback is the generation of network traffic overhead.

In order to avoid congestion due to diagnostic messages, in many cases this health status information is gathered and dispatched along a secondary network. This is evolving towards dedicated ancillary subsystems—an example of this trend is the Intelligent Platform Management Interface (IPMI); a complete fault tolerance solution for HPC clusters that uses IPMI is FTB-IPMI [11].

3. Local fault monitor (LO|FA|MO) approach

LO|FA|MO is a design approach that provides systemic fault awareness in a distributed system. A LO|FA|MO-enabled system requires:

- a dedicated hardware block inside the network interface card hosted on each node that implements the 3D torus network topology;
- a dedicated software component running on each system node;
- a 3D torus network connecting computing nodes;
- a Service Network for diagnostic messages.

A mutual watchdog mechanism between the two LO|FA|MO components ensures that the health status of the host and the NIC is monitored on each node (locally). The 3D torus network and the service network are two redundant paths for diagnostic messages about fault and critical events occurring on each node; moreover, the 3D toroidal network topology allows each node to monitor its first neighbours. Diagnostic messages are delivered to a few selected nodes (Supervisor Nodes, or shortly Supervisors) that gather information about faults occurring to the whole system, thus composing the big picture. The overall structure is clearly hierarchical (Fig. 1): faults can be detected locally on each node (lower level), or by first neighbours nodes (mid level); Supervisor