EURO-PAR
CONFERENCE 2019

ABSTRACTS RESILIENCE

SPACE-EFFICIENT REED-SOLOMON ENCODING TO DETECT AND CORRECT POINTER CORRUPTION

Scott Levy and Kurt B. Ferreira

Concern about memory errors has been widespread in high-performance computing (HPC) for decades. These concerns have led to significant research on detecting and correcting memory errors to improve performance and to provide strong guarantees about the correctness of the memory contents of scientific simulations. However, power concerns and changes in memory architectures threaten the continued viability of current approaches to protecting memory (e.g., Chipkill). Returning to a less protective error-correcting code (ECC), e.g., single-error correction, double-error detection (SECDED), may increase the frequency of memory errors, including silent data corruption (SDC). SDC has the potential to silently cause applications to produce incorrect results and mislead domain scientists.

We propose an approach for exploiting unnecessary bits in pointer values to support encoding the pointer with a Reed-Solomon code. Encoding the pointer allows us to provides strong capabilities for correcting and detecting corruption of pointer values. In this paper, we provide a detailed description of how we can exploit unnecessary pointer bits to store Reed-Solomon parity symbols. We evaluate the performance impacts of this approach and examine the effectiveness of the approach against corruption. Our results demonstrate that encoding and decoding is fast (less than 45μs per event) and that the protection it provides is robust (the rate of miscorrection is less than 5% even for significant corruption). The data and analysis presented in this paper demonstrates the power of our approach. It is fast, tunable, requires no additional per-pointer storage resources, and provides robust protection against pointer corruption.

IMPROVING RELIABILITY FOR PROVISIONING OF VIRTUAL MACHINES IN DESKTOP CLOUDS

Carlos E. Gómez, Jaime Chavarriaga, Harold E. Castro, and Andrei Tchernykh

Desktop clouds (DC) provide services in non-stationary environments that face reliability and performance threats not found in traditional clusters and datacenters. The idle resources available on computers can be claimed by users, turned off and faulted any time. For instance, platforms such as CernVM and UnaCloud harvest idle resources on computer labs to run virtual machines and support scientific applications. This non-stationarity is one of the main sources of issues in the design of reliable desktop cloud infrastructures that are capable of mitigating their own faults and errors. Based on a fault analysis that we have been carrying out and refining for a couple of years, we have found that reliability problems begin as the number of virtual machines that are going to be executed increases; these virtual machines must first be provisioned in the physical machines where they will be hosted. On the one hand, the main factors that can affect the provisioning of virtual machines in a DC are: the use of disk space, and the transmission of virtual images over the network. On the other hand, the applications and actions performed by users in the desktops may cause the virtual machine malfunction. In this paper, we propose an appropriate strategy for the scalable provisioning of VMs, describe the implementation, and analyze its effectiveness.

PHYSICS-BASED CHECKSUMS FOR SILENT-ERROR DETECTION IN PDE SOLVERS

Maher Salloum, Jackson R. Mayo, and Robert C. Armstrong

We discuss techniques for efficient local detection of silent data corruption in parallel scientific computations, leveraging physical quantities such as momentum and energy that may be conserved by discretized PDEs. The conserved quantities are analogous to "algorithm- based fault tolerance" checksums for linear algebra but, due to their physical foundation, are applicable to both linear and nonlinear equations and have efficient local updates based on fluxes between subdomains. These physics-based checksums enable precise intermittent detection of errors and recovery by rollback to a checkpoint, with very low overhead when errors are rare. We present applications to both explicit hyperbolic and iterative elliptic (unstructured finite-element) solvers with injected memory bit flips.

CHECKPOINTING KERNEL EXECUTIONS OF MPI+CUDA APPLICATIONS

Max Baird, Sven-Bodo Scholz, Artjoms Šinkarovs, and Leonardo Bautista-Gomez

This paper proposes a new approach to checkpointing MPI applications that use long-running CUDA kernels. It becomes possible to take snapshots of data residing on the GPUs without waiting for kernels to complete. The proposed technique is implemented in the context of the state of the art high performance fault tolerance library FTI. As a result we get an elegant solution to the problem of developing resilient MPI applications where GPU kernels run longer than the mean time between hardware failures. We describe in detail how we checkpoint/restart collaborative MPI-CUDA applications, and we provide an initial evaluation of the proposed approach using the Livermore Unstructured Lagrangian Explicit Shock Hydrodynamics (LULESH) application as a case study.

Submission deadline for the Euro-Pas PhD Symposium has been extended to 27 May, 2023. Click here for more information - https://t.co/wWxisICJSC

17.05.2023 - 11:51

The Euro-Par PhD Symposium is a welcoming and supportive forum for PhD students to present their work. Click here for more information: https://t.co/wWxisICJSC

04.04.2023 - 09:25

Submit your paper for EURO-PAR 2023 Workshops and Minisymposia! ⬜Click here for more information. https://t.co/UEseXWb3Dz

07.03.2023 - 08:18

Abstract submission is due tomorrow 24 Feb, 2023 ⬜ https://t.co/eH2C9CRZA3

23.02.2023 - 0

CONTACT US

HOSTS

GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

GWDG
Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen

SUB | NIEDERSÄCHSISCHE STAATS- UND UNIVERSITÄTSBIBLIOTHEK GÖTTINGEN

SPONSORS

Atos

Springer Link

DDN
AI · BIG DATA · HPC

intel Software

ho COMPUTER
www.hocomputer.de

intel Software
Elite Reseller

SHARE ON: