2021 年 OS 研　目次案

井口 卓海

2021 年 4 月 26 日

目次

1 はじめに

- メモリはページテーブルなどの重要なデータ構造を配置しており，メモリにおける故障の発生はOSにとって致命的な影響を及ぼす
  - メモリエラーにより，kernel panicなどのシステムを停止・再起動しなければならない状況になる
  - メモリエラーにはbit反転のようなソフトエラー，物理メモリの破損のようなハードエラーがある
  - メモリ故障の修正機能を持つECCのようなメモリも存在するが，修正できるのはソフトエラーのみ
- ヘテロジニアスメモリシステムを導入することにより，データの重要度に応じた配置でメモリエラーへの耐性を向上させる
  - ヘテロジニアスメモリシステムとは，性質の異なるメモリを組み合わせて構築するメモリシステムのこと
    - メモリの性質とは，エラー耐性のような信頼性，データI/O速度のような性能，消費電力等が挙げられ，メモリの種類により一長一短
  - ハードウェアの領域ではDRAMとECCのような信頼性の異なるメモリを組み合わせてメモリシステムを構築可能
- 既存OSはデータの重要度に応じたデータ配置をせず，ヘテロジニアスメモリに対応できない
  - 高信頼メモリは容量が少ない傾向にあり，どのデータを高信頼メモリに置くべきか選別する必要がある
    - データの重要度・破損時の回復可能性・破損時のシステム全体への影響を鑑みることが考えられる
  - 既存OSは単一の物理メモリで構成されているという前提のもとデータを配置しており，データ・メモリの性質に応じた配置をしていない
  - ヘテロジニアスメモリシステムを導入しても，データを狙い通りに配置することができない
- メモリエラーに対して耐性を持つOSを提案する
  - メモリ側で修正不可能なエラーを検知しても，システム全体のダウンを回避して動作継続を可能にする
  - ECCのようなメモリにおいて修正できないメモリエラーを検知した場合，エラー箇所に応じたリカバリハンドラによる回復を実施し，局所的なメモリ破損を抱えた状態でも即座のシステムダウンに繋げず動作を継続させる
  - メモリ上に置かれ，役割ごとにデータを管理するメモリオブジェクトの単位

でリカバリハンドラを呼び出し，無事なデータの退避，破損箇所に対応するデータの初期化，整合性の保持を行う

- 本論文での貢献
  - ECC のような高信頼メモリでも修正不能なメモリエラーに対して耐性を持つ OS を提案した．OS 自体にメモリエラーの耐性を持たせることで，メモリと合わせてさらなる可用性の向上と，低信頼性のメモリを使用した場合においてもメモリエラーによる悪影響を軽減できると考えられる (3 章).
  - 提案手法を実現するため，OS のデータをメモリオブジェクトの単位でデータのセマンティクス・使用される状況・データ構造間の関係性・回復可能性を明らかにし，設計を行った．故障が発覚したアドレスが渡されるという仮定の下，アドレスから故障したメモリオブジェクトを探索，合致したリカバリハンドラを呼び出すように実装した (4 章).
  - MIT の開発した小規模な教育用 OS である xv6 に対して実装を行った．想定したクラッシュシナリオ上において，システム全体がダウンすることなく継続して動作することを確認した (5，6 章).

2 背景

2.1 メモリエラー

  - ビット反転によるメモリエラー
  - 物理的破損によるメモリエラー
  - 実環境におけるメモリエラーの発生状況
- メモリの種類
  - DRAM，ECC 等
  - ビット反転の修復方法

2.2 ヘテロジニアスメモリシステム

- ホモジニアスメモリとヘテロジニアスメモリ
- HW におけるヘテロジニアスメモリの扱い

2.3 関連研究

- メモリエラー耐性の指標と分析

  - Characterizing Application Memory Error Vulnerability

- 性能向上を目的としたヘテロジニアスメモリの利用
  - HeteroVisor
  - HeteroOS
  - KLOCs

2.4 問題点

- これらはサーバ向けであり，OS を対象としていない
- OS 自体がヘテロジニアスな構成を認識しておらず，VM を介さない一般的なコンピュータの場合にはヘテロジニアスな恩恵を享受できない

3 提案

- 提案
  - 本研究では，メモリ側で修正不可能なメモリエラーが発生した際に，リカバリを実施することでシステム全体のダウンを回避し，継続動作を可能にする OS を提案する
- 想定する状況
  - メモリエラーが発覚した箇所のアドレスは，そのメモリにアクセスした時点で割り込みにより OS 側に知らされる
    - ECC メモリを使用する想定
  - メモリエラーはメモリオブジェクト単位で 1 つのみ発覚し，破損したメモリオブジェクトはその全メンバが使えない
    - メンバに破損したメモリオブジェクトを持つ場合は除く
- アプローチ
  - メモリオブジェクトのアドレスをまとめたリストを事前に用意し，破損が発覚したらリストを引いて破損したメモリオブジェクトを特定
  - メモリオブジェクトのセマンティクス・使用状況・他メモリオブジェクトとの関連に応じたリカバリハンドラにより復旧を試みる
- デザインチャレンジ

  - どうやって破損が検知された箇所のアドレスから対応するメモリオブジェクトを特定するか
  - リカバリをどのように実施するのか

4 設計

4.1 アドレスリスト

- アドレスリストの作成
  - アドレスリスト専用の構造体を用意し，メモリオブジェクト生成時等のタイミングで先頭アドレスを登録
  - メモリオブジェクトによってはアドレスリストが準備できた段階で登録
  - メモリオブジェクトが free 等で使われなくなった際にはリストから削除

4.2 アドレスリスト探索・ハンドラ仲介関数

- 仲介関数においてメモリから受け取ったアドレスでリストを探索し，破損したオブジェクトを特定
- 特定したオブジェクトをリカバリするためのハンドラを呼び出して回復を試みる

4.3 リカバリハンドラ

- 受け取ったアドレスからアドレスリストによって破損箇所を特定
- 新たに領域を割り当て，破損したメモリオブジェクトの代替物を用意して無事なデータを移動
- 破損したメモリオブジェクトに関連するその他のデータとの整合性を保持

5 実装

5.1 アドレスリストの作成

- 空きページを管理するオブジェクトのような，数が非常に多いオブジェクトは 1 ページ内にノードを複数埋める
- メモリオブジェクトの登録だけでなく，削除・再登録に対応

5.2 アドレスリスト走査・ハンドラ仲介関数

- メモリオブジェクトの種類によっては，他のリカバリハンドラが動かせない状況になるため，リストを探索する順序を調整
- マルチプロセスを使用している際等，二重にリカバリハンドラを呼び出す場合に対処

5.3 リカバリハンドラ

- 各メモリオブジェクトのリカバリハンドラについて記述 or 工夫を凝らしたものやわかりやすいものについてのみ記述

6 実験

6.1 目的

6.2 実験環境

6.3 クラッシュシナリオ

6.3.1 クラッシュシナリオ 1

6.3.2 クラッシュシナリオ 19

6.4 クラッシュシナリオ (マルチプロセス)

6.4.1 クラッシュシナリオ 1

7 おわりに

- まとめ
- 課題
  - ページテーブルへの対応について