たたかえ！オープンフォース #45

世界を護る葉々

～ぼくのかんがえたさいきょうのぶんさんアーキテクチャ～

2013.11.23 オープンセミナー徳島 LT

秘密結社オープンフォース総統　河野

自己紹介

- 秘密結社オープンフォース
  - 秘密結社
    - 破壊活動を
  - オープン
    - オープン技術で
  - フォース
    - 世界を変える

PCの終焉

- 管理コストの増大
- スマホ、タブレットへ

管理コストとは？

- セキュリティ
- しろうと（≒ユーザ）がセキュリティを担保できない
- ピー―ピーの[redacted]設ピーがピーであるが
- それが本質ではない

本質的な解決ではない

- スマホ、タブレットも
- 利便性のために
- セキュリティ担保が破れている

セキュリティルールで担保

- プログラムは特定の条件下でしか実行できない
- 単純明快なルールを徹底するだけで
- 根本的な改善
- CF.,SQUARANTINE

ネットワーク接続

- ネットサービスに接続するオフィス機器
- ネット家電
- ネットガジェット
- ウェアラブル

- TCP/IPが必須へ

ネットワークに接続するアイテムを軽量に開発できない

- 他の面で開発に必要なリソースは
- 軽減しているのに
- セキュリティがネック
  - セキュリティホール
  - メンテナンスが必須

セキュリティホール

- ロジックの作り込みの不良
  - インジェクション
- 設定不良
  - ディレクトリリスティングを許す
- ソーシャル介在
  - メールバウンド
  - ストレージバウンド
- メモリ破壊
  - バッファオーバーラン
  - スタックアンダーフロー

セキュリティホール

- これらは各種ルールの導入で防げる
  - ロジックの作り込みの不良
    - インジェクション
  - 設定不良
    - ディレクトリリスティングを許す
  - ソーシャル介在
    - メールバウンド
    - ストレージバウンド

- 防げない
  - メモリ破壊
    - バッファオーバーラン
    - スタックアンダーフロー

メモリ破壊

- OSレベルにも関係
- スタック、ヒープを使っている以上無理
- そもそも、スタック、ヒープって必要なのか

本質的にクラッシュの問題点

- ヒープ
- 動的メモリ割り振り
- バッファオーバー
- スタック

ヒープ

- 悪影響
  - GC
    - ガッページコレクション

＿＿人人 人人＿＿
＞ 突然のGC ＜
￣￣Y^Y^Y^Y￣￣

  - リアルタイム処理のネック
  - GCの処理をどうするか
  - ヒープ無くせばリアルタイム可能では？

ヒープを無くすには？

- 静的にとる
- 再入の禁止、または再入回数の制限

動的メモリ割り振りをやめるには？

- オブジェクト指向を使わない
- CALLの制限

バッファオーバーを無くすには?

- 配列、ポインタに起因
- ハードウェアによる監視（ランタイム）
- 添字チェックルーチン（ランタイム）
- コンパイラによる事前チェック

コンパイラによる事前チェック

```
width = GetPictureWidth(“sample.jpg”);
Height = GetPictureHeight(“sample.jpg”);
Count = 0
For ( x=0, x<width, x++)
{
    For ( y=0, y<Height, y++)
    {
        If ( 0 == GetPicturePixell(x,y) )
                {
                    workBuffer[count++]=x;
                }
    }

}
```

- 画像ファイルのブランクピクセルのX座標をリストアップする

コンパイラによる事前チェック

```
width = GetPictureWidth(“sample.jpg”);
Height = GetPictureHeight(“sample.jpg”);
Count = 0
For ( x=0, x<width, x++)
{
      For ( y=0, y<Height, y++)
      {
            If ( 0 == GetPicturePixell(x,y) )
                  {
                        workBuffer[count++]=x;
                  }

      }

}
```

- WORKBUFFERの添字
- 最大値はINT型のMAXの二乗
- 常に最大値を予測すると非実用的
- この場合はWIDTH、HEIGHTの最大値を事前に設定
- 添字の最大値を超えた場合の処理ルーチンの設置を義務付け

スタックを無くすには？

- スタックの代わりに静的メモリを使うとすると
  - スタック上への動的メモリ確保
  - CALL
  - タスク切り替え
  - 割り込み

スタック上への動的メモリ確保

- しない→静的確保へ

CALL

- リエントラントしない
- CALLの深さが予測可能になる
- 高級言語による予測
- 事前管理して必要な量を静的割付

タスク切り替え

- タスク数上限の設定
- メモリ量内に収まる量だけ可能

割り込み

- 多重割り込みの深さ
- 同時割り込みの数
- 制限してメモリ利用量を確定

メモリ破壊

- 配列や添字
- 高級言語を前提
- 先のヒープ対策と同じ戦略

メモリ多め

- オーバーヘッドが無い
- GCの心配がない

開発をどうするか

- 高級言語の開発？
- 再入の制限
- CALLの制限
- 処理深さの予測

実現できるのか

- Lチカ　OK

```
LOAD  1→ A        // 1 is OutputPortNo.
:ON
OUT [A],1         // LED ON
LOAD 0 →B         // Counter Clear
:LOOP1
SLEEP 1           // 1ms wait
ADD [A],1
CMP 1000,[B]
IfJumpEqale :OFF
Jump     :LOOP1
OUT [A],0         // LED OFF
:LOOP2
LOAD 0 →B         // Counter Clear
SLEEP 1           // 1ms wait
ADD [A],1
CMP 1000,[B]
IfJumpEqale :ON
Jump     :LOOP2
```

TCP/IPスタック

- 単純化して1スレッドの処理のみ
- MTU値のみバッファを持てば良い
- CF.,COBALTBLUE

COBALTBLUE

- ANDROIDのBLUETTOTHテザリングアプリ
- TAMさん作成
- JAVAアプリ空間で書かれたTCP/IPスタック
- 変数はほとんど静的
- フットプリント数十KBYTES
- →可能？

TCP/IPの喋れる

- 軽量な処理ができるだけで

波及効果

- APACHEモジュール
- 組込み
- ネットガジェット
- 分散処理

分散処理

- 単機能のノード
- たくさん束ねる
- ネットワークで結合

単機能のノード

- CPUパワー小
- メモリ大
- レスポンス大

CPUパワー小

- 16BIT程度でも可能
- 組込み用のCPUが使用できる

組込み用のCPU

- 安価
- 省電力
- アクティブ冷却が必要ない

メモリ大

- とはいえ
- 半導体製造プロセスの進歩
- CF,RASPBERRY PI
  - ワンチップで512M

- もっと少なくてもよければ不揮発性メモリが使える
- →圧倒的な省電力の実現へ

レスポンス大

- 反応時間の正確な予測が可能
- 分散処理管理の低減

1000ノード

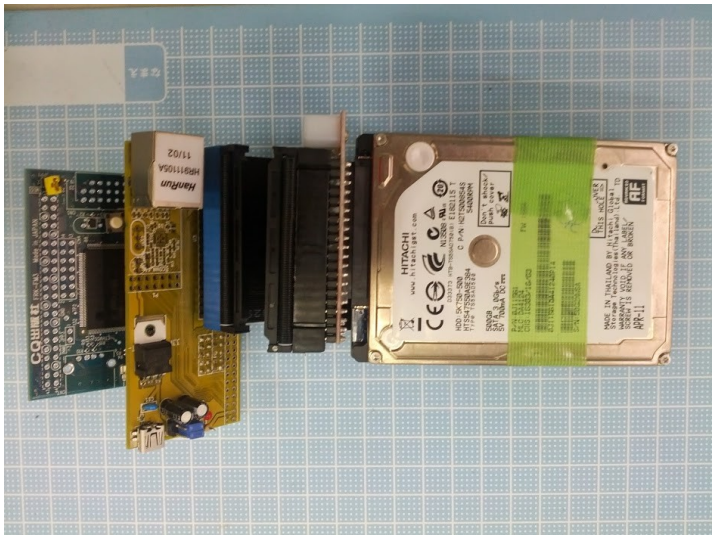
- HADOOP由来の分散処理システム
- 64MBYTESを1単位として
- 512GBYTESの補助記憶装置がつく

現在

- ハードウェアの設計
  - 試作
    - 2013.4.6 OSC 2013 TOKUSHIMA
- 分散フレームワーク
  - HADOOP由来のものより要件を作成
    - 2013.6.17 第5回分散処理勉強会
- ファームウェア
  - 言語の要件検討  今ココ！
    - 2013.11.23 オープンセミナー2013@徳島 LT
    -

ハードウェア

- 分散処理ノード用に開発
- ネット家電やガジェットに気軽に転用できる



- セキュリティホールフリーへ

データセンターの建設へ

- エネルギーがネック
  - エネルギーコスト
  - 熱密度
- コストも下がり、
- 発熱が少ない
  - 高密度

データセンター

- 今後20年間のテクノロジーエンジン
- 国家
- 世界
- 発展の鍵をにぎるもの

更に未来へ

- サイバー戦
- 1秒ごとに新たな脅威
- 現状のシステムの延長線では免疫不全に
- 飛躍のネックを乗り越えて

俺たちの戦いはこれからだ！

次回作にご期待ください