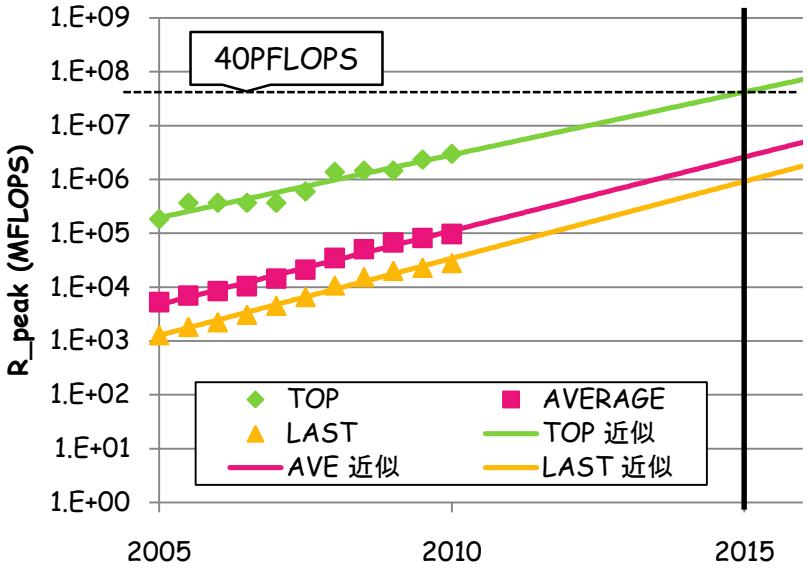
5年後のジョブ耐故障化ミドルウェア事情

實本　英之

5年後何がスタンダードなのか？

- 性能
  - 40PFLOPS



- 同時利用プロセス数
  - 3PFLOPS → 1.3M (2010 Y)
  - 単純比例と考えるなら… 40PFLOPS → 18M
  - 運用時の最大数を考えると
    - 15K - 30K コア級のマシンでおおよそ1024コアの同時利用プラン
    - 100Kプロセス単位？

100Kプロセスとは

- 並列化効率
  - 50%の性能向上率を得たい場合
    - プログラムの並列化不可能部分が 0.001 % 以下
- 故障率
  - 1プロセスが10年に1度落ちるとしても
    - 30秒に1回どこか故障する！！

（実は3カ月に1回くらいなんだかんだで落ちたりするわけだが…）

と脅しまくるのが今までの考え方だったのですけれども…

ミドルウェアによる(ジョブ耐故障化の)真髄?

- 並列化効率の低いアプリケーション
  - たくさんのプロセスが使えない
  - 台数増加による故障率上昇が低い(現状並み)

- 並列化効率の高いアプリケーション
  - プロセス間イベントが少ない
  - ジョブ全体への故障波及が少ない
  - 故障グループ内のプロセス数が少ないので部分的な故障率が低く抑えられる

アプリケーションに逆らうな！！
→だけだと微妙なので

現在のHPCにおけるジョブ耐故障化

- Coordinated Checkpointing
  - チェックポイント(プロセスイメージのスナップショット)を定期的にとることによるプロセス状態の冗長化手法
    - 予約したノードをすべて計算に利用できる
  - すべての構成コンポーネント間で同期をとりコンポーネント間の一貫性を確保
    - コンポーネント間イベント数に耐故障化コストが依存しない
- 並列化効率がそれなり、コンポーネント間依存性も高いアプリケーション向け
- チェックポイント時間が増大してきているので小手先の削減技術

これからの耐故障性

- 複数のコンポーネント間の依存性に応じた適切な手法を用いる
  - Coordinated
    - コンポーネント間イベント数にコストが依存しない
    - コンポーネント間の依存性を高める

    →並列化効率の低いアプリケーション向け
  - Uncoordinated
    - コンポーネント間の依存性が低い
    - コンポーネント間イベント数にコストが依存する

    →並列化効率の高いアプリケーション向け

  - Coordinated + Uncoordinated な手法
    - コンポーネントをいくつかのグループに分ける

極論を言うと

- 環境に合わせてすべてのアプリケーションが高い効率化性能を持つようになる
  - コンポーネント間イベントが皆無に…？
  - 並列・耐故障化のアルゴリズムなんてほとんどいらない
  - 逐次・耐故障化をどれだけ早くできるか
    - 小手先だった対処策のはずが本質へ

      →ローカルノードへのチェックポイント保存と Erasure Coding

      →チェックポイント保存データの削減

      →チェックポイント保存タイミングの調整

運用における
チェックポイント／リスタートの最適化

- 実は本気で耐故障機能を含んだHPCシステムは運用されてないのでは？？？
- 運用スパコンでは、リスタートプロセスにノードがアサインされるまで時間がかかる
  - 同時利用プロセス数に合わせた、冗長ノードを無料アサインする

    →冗長ノードは無保証ジョブ用のノードとして使う？

    故障が起こったら、無保証ジョブは強制ストップ
  - ユーザに故障復旧分のノードをアサインさせる

故障復旧を見込んだ
スケジューリングポリシがほしい

チェックポイントからレプリケーションへ

- 超大規模化による利用スタンスの変化
  - アプリケーションの並列限界がある
  - 故障による時間損失に比べたら冗長ノードを確保しておくほうが安くつくかも？

- レプリケーション
  - コンポーネント間イベントを冗長コンポーネントへフォワードする
  - Periodic なチェックポイントはコストが大きい
    - ファイルの保存、転送
  - 故障への対応回数に制限がある
    - コンポーネント状態の複写ができれば対応可能

    →fork/migration とかシリアルチェックポイントの併用とか

まとめ

- アプリケーションの性質に抗わない耐故障機能を実現できないとだめ
  - 単純に環境規模で故障率は決まらず、アプリケーションの性質に依存する
  - 並列化効率の高いアプリケーションの邪魔をするような耐故障機能はこれからの超大規模環境に向かない
- 耐故障機能の本格運用はこれから始まる
  - 復旧ノードのスケジューリング等はあまり考えられていない
  - 利用コストに依存したノード予約最適化
    →冗長ノードの確保も場合によってはあり
- 実は故障検知も重要な問題なのですが…
  時間がありませんでした。

（おまけ）チェックポイントも早いといいなぁ

- やっぱり冗長ノードはいやだなぁと。
- MRAM と ハードウェア支援でどうにかならない？（5年のスパンでは無理だけど）
  - 本体OSや一部のHWが死んでても外部からアクセスできる特殊なメモリがあれば…
    - ハードウェアによる転送のサポートとか…
    - NICからメモリまで電源供給とか…
    - MRAM は不揮発性メモリなうえDRAMのように使える
  - forkするだけでチェックポイント完了？
    - ページテーブルのコピーだけで軽そう
    - forkプロセスはsigwait しておくだけでOK
- 2010/4 の段階で 16Mbit 出荷….
- 2010/5 の段階で 5Gbit 級の算段がついた