【B4】Delphi/C++テクニカル/テクノロジートレンド

EMBARCADERO TECHNOLOGIES

DEVELOPER CAMP

Delphi／C++Builderで実践する並列プログラミング ――マルチコアからクラスタへ

株式会社イマジオム 代表取締役
高木太郎

アジェンダ

EMBARCADERO TECHNOLOGIES DEVELOPER CAMP

- 第1部：並列処理の基礎
  - 並列処理とは何か
  - マルチCPU、マルチコア、クラスタ、グリッド
  - 要素技術
    - 通信、マルチスレッド、ノード管理、データ管理、スケジューリング
  - 性能
    - フォーク・ジョインモデル、粒度、アムダールの法則、最速ノード数
  - 既存のテクノロジ
- 第2部：実践、クラスタプログラミング
  - PCクラスタミドルウェア「HarmonyCalc」
  - 例題「マンデルブロ集合」

はじめに

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- このセッションを準備するにあたって悩んだこと――
  当社製品「HarmonyCalc」の使用を前提にお話しするべきか？
  - 特定の製品に依存しない一般的な内容にするか？
  - 具体的・実践的で、応用しやすい内容にするか？

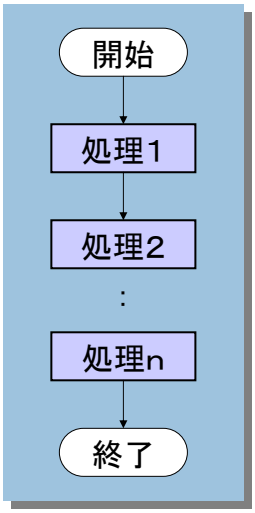
- 第2部では、HarmonyCalcの使用を
  前提にお話しすることにしました。
  その代わり……

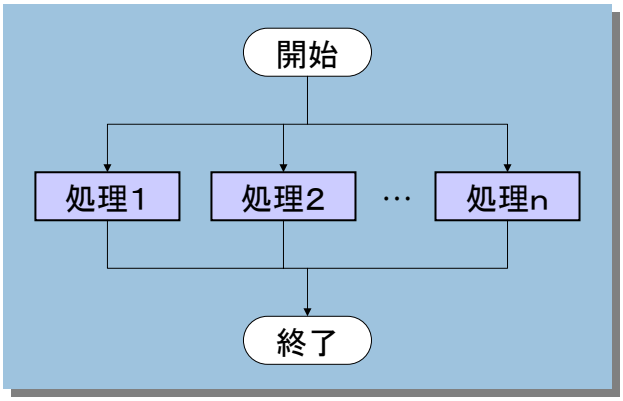
EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

第1部：並列処理の基礎

並列処理とは何か

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

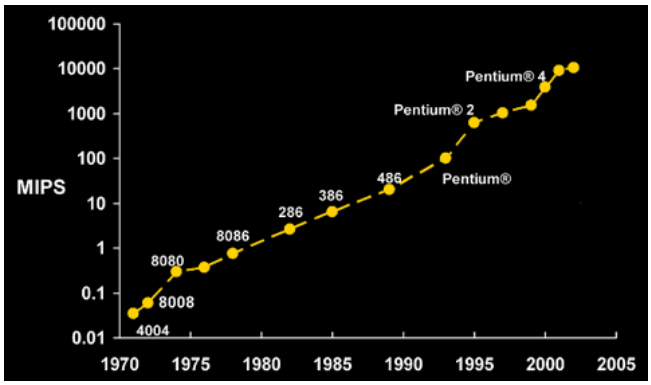




並列処理＝処理を分割し、複数のプロセッサで同時に実行すること

並列処理の必要性

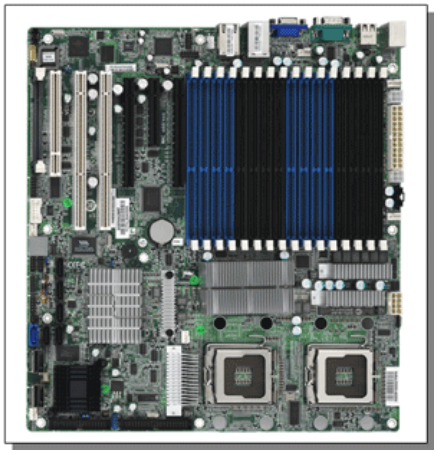
EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

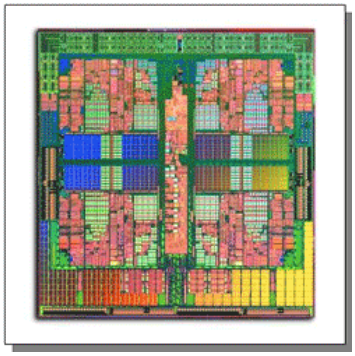


「ムーアの法則」の限界に達したら、並列化しか道はない

並列化(1):マルチCPU・マルチコア

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

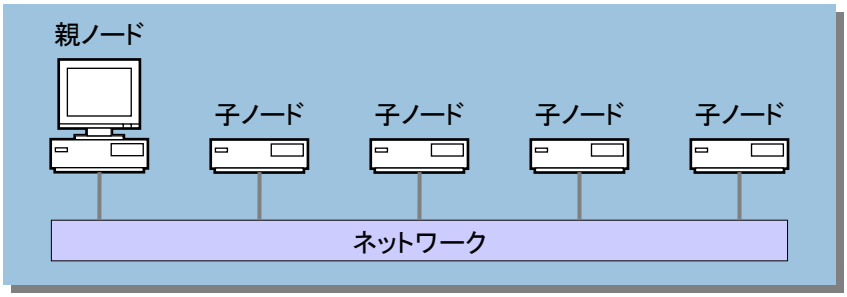




シングルコアから
メニーコアへ

並列化(2):クラスタ・グリッド

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP



ラックから
一体型へ

混同されやすいクラスタとグリッド

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- 「集中管理」と「分散管理」の違いで、まったく異なる方向に

	クラスタ	グリッド
ハードウェア管理者	少数	多数
OS	Linux・Windowsなど	Windowsに限られる
通信ハードウェア	Ethernetの他にInfinibandやMyrinetも	Ethernetに限られる
通信プロトコル	伝送速度優先	信頼性・守秘性・汎用性優先

並列処理技術の使い分け

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

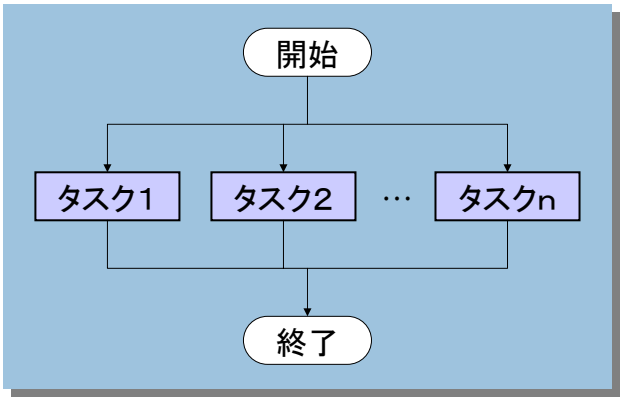
- 「現在の処理時間」と「必要な処理速度」で技術が決まる

現在の処理時間	必要な処理速度		
	2～20倍	20～1000倍	1000倍以上
1分	マルチCPU マルチコア	クラスタ	（実現不可能）
1時間			クラスタ
1週間			
1年			グリッド

フォーク・ジョインモデル

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- 分岐（fork）と統合（join）の組み合わせが並列処理の基本
  - 「タスク」（task）：forkとjoinの間で並列実行される処理
  - 「粒度」（granularity）：タスクの（平均的な）処理量



性能（1）――アムダールの法則

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- アムダールの法則（Amdahl's law）：
  アルゴリズムが「並列化に適しているか」を判定する法則

$$a = \frac{1}{\frac{p}{n} + (1 - p)}$$

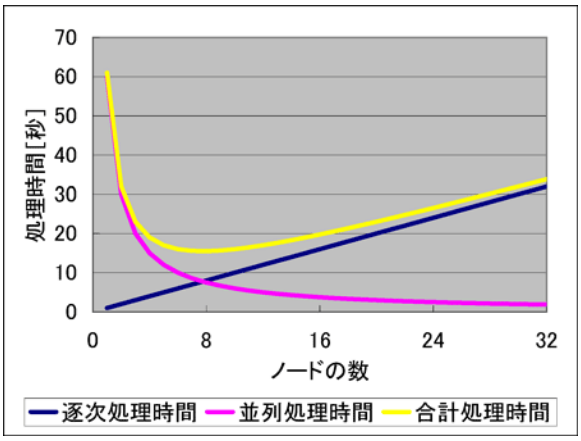
a	処理速度倍率
n	プロセッサの数
p	逐次処理部分の割合

- タスク分割のできない「逐次処理部分」の割合が、最大処理性能を決める

性能(2)――最速ノード数

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- ノードの数を増やすほど、処理時間が短くなるとは限らない



並列処理を実現する要素技術

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- 並列処理は、いろいろな要素技術の組み合わせ
  - 通信(communication)
  - マルチスレッド(multi-threading)
  - ノード管理(node management)
  - データ管理(data management)
  - スケジューリング(scheduling)

要素技術(1)——通信

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- クラスタ・グリッドでは、コンピュータ間でのデータ交換が必要
- プロトコルをどのように設計するか？
  - クラスタ:伝送速度を重視
  - グリッド:信頼性・守秘性・汎用性を重視
- どのような通信ハードウェアを使うか？
  - トレンド＝特殊なハードウェアから、一般的なハードウェアへ
  - 通常はギガビットEthernetとIPプロトコルで充分

要素技術(2)——マルチスレッド

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- マルチCPU・マルチコアの性能を引き出すには、マルチスレッドプログラミングが必要
- マルチスレッドの問題をどのように避けるか？
  - 同期、排他制御
  - デッドロック・スタベーション
- ソースコードの可読性をどのように保つか？
  - 多くの開発言語では、別スレッドでの実行をインラインに記述することができない

要素技術(3)——ノード管理

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- クラスタ・グリッドでは、システム構成の定義と、システム全体のリアルタイム監視が必要
- システム構成をどのように記述するか？
  - 単純な記述
    - ノードを収容するコンピュータ
    - ノードへのアクセス方法
  - 高度な記述
- 故障ノードをどのように検出するか？
  - TCPの送信タイムアウトによる検出
  - UDP／ICMPパケットの不着による検出

要素技術(4)——データ管理

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- データへの同時アクセスによる障害や、データの消し忘れによるリソースリークを避ける仕組みが必要
- 同時アクセスをどのように回避するか？
  - アクセスを時間的にずらす：排他制御
  - アクセス対象を分離する：一時データの作成
- リソースリークをどのように避けるか？
  - データのコピー・消去を追跡していれば、処理中断時やエラー発生時のリソースリークが避けられる

要素技術(5)——スケジューリング

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- 個々のコンピュータやノードを協調させて一つの有用な処理を行わせるため、リアルタイムな動作計画が必要
- ノードとのコミュニケーションをどのように設計するか？
  - 指示:スケジューラからノードへ
  - 通知:ノードからスケジューラへ
- どのようなスケジューリング戦略が優れているか？
  - 手の空いたノードに、どんどん処理を割り振っていくのが基本
- OSの性能を高めるために何ができるか？

並列プログラミングの性格

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- ハードウェアの個性を活用した方が高性能
  - オブジェクト指向との相性が必ずしもよくない
- 大きい処理単位(粒度)で並列化した方が高性能
  - アプリケーション開発者自身の努力が必要
- 「性能」と「使い勝手」を折り合わせる考え方によって、いろいろなテクノロジが乱立し、なかなか統一されない。
  - スレッドAPI(thread application program interface)
  - MPI(message passing interface)
  - OpenMP(Open Multi-processing)
  - 並列化ライブラリ(parallelized libraries)

既存のテクノロジ（1）－スレッドAPI

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- マルチスレッドプログラミングのため、OSが用意している機能
- OSに付属しているので、OSを持っていれば誰でも使える

長所	OS以外のものが不要 開発に必要な情報が入手しやすい
短所	スケジューラをプログラムに実装する必要がある ノード・データをプログラムで管理する必要がある クラスタに対応していない マルチスレッドに固有の問題が起こりやすい ソースコードの可読性が失われやすい

既存のテクノロジ（2）－MPI

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- OSの扱うメッセージを、他のコンピュータに伝送するための インタフェース
- オープンソースとして標準化が進み、各種のハードウェアに対応

長所	性能向上のためのチューニング自由度が高い
短所	スケジューラをプログラムに実装する必要がある ノード・データをプログラムで管理する必要がある プログラミングが非常に複雑 プログラムのテスト・デバッグが難しい マルチスレッドに固有の問題が起こりやすい ソースコードの可読性が失われやすい

既存のテクノロジ(3)－OpenMP

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- 並列コンパイラのための拡張言語仕様
- C／C++、Fortranに対応
- オープンソースとして標準化が進められている

長所	プログラムのテスト・デバッグが簡単 ソースコードの可読性がほとんど失われない
短所	限られた言語・コンパイラしか対応していない クラスタへの対応が不充分 データをプログラムで管理する必要がある ファイル形式のデータに対応していない

既存のテクノロジ(4)－並列化ライブラリ

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- 長い時間のかかる汎用的な処理を、マルチCPU・マルチコアのために並列化したライブラリ
- いくつかのベンダが製品として販売

長所	並列プログラムを自分で開発する手間が省ける
短所	必要な機能が手に入るとは限らない 大きい粒度での並列化が難しく、高性能が得にくい クラスタへの対応が不充分 市販の製品は一般にかなり高価

EMBARCADERO TECHNOLOGIES

DEVELOPER CAMP

第2部：実践、クラスタプログラミング

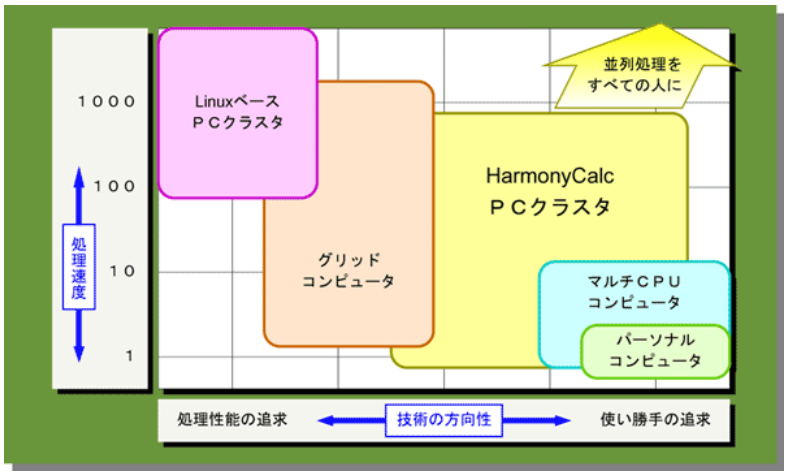
PCクラスタミドルウェア「HarmonyCalc」

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- PCクラスタミドルウェア
  - アプリケーションをPCクラスタで動作させるための組み込みソフトウェア
- 自社製品の画像処理・図形処理プログラムを高速化するために開発した要素技術を製品化
  - ユーザの視点・論理で設計
- オリジナリティ＝「使い勝手」を最も重視
  - 開発しやすさ
  - 保守しやすさ
  - 配布しやすさ

「並列処理を、すべての人に」

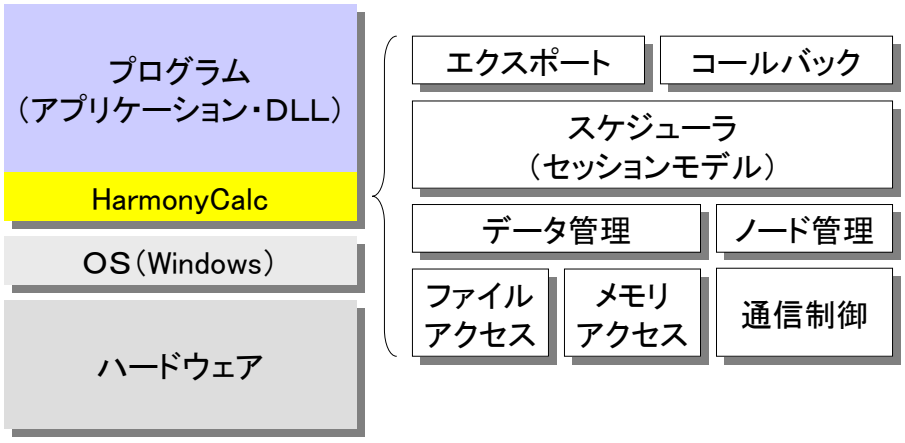
EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP



HarmonyCalcの構成

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- 五つの要素技術をDLLに実装



セッションモデルとセッション

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

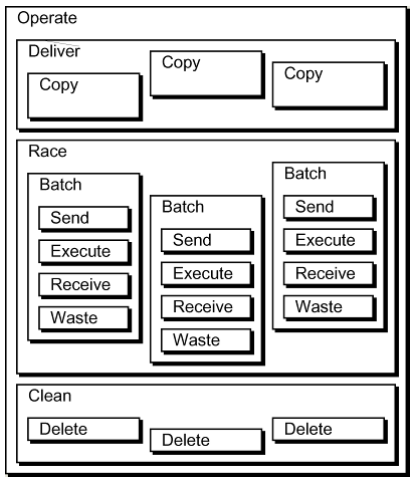
- 「セッションモデル」＝HarmonyCalcの並列プログラミングモデル
  - セッション階層：並列処理手順を「セッション」の階層として一般化
  - イベントドリブン：処理をブロックし、スケジューリングを独占的に実行
  - 仮想ノード：実際のハードウェア構成に関係せずにノードを定義
  - 独占的なデータ操作：データのコピー・消去をプログラムに行わせない
- 「セッション」＝並列処理の構成要素
  - 包含関係と前後関係、関与するノードを規定
  - 開始時と終了時にイベントを発生させる

→ ハードウェア構成をプログラムから隠蔽

並列処理手順の一般化

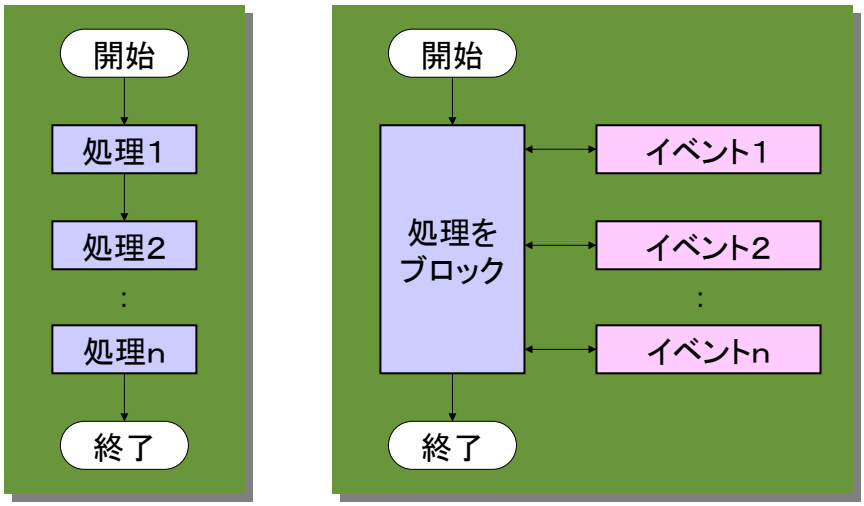
EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- 11種類のセッションの組み合わせで並列処理手順を記述
  - コピー（copy）セッション
  - 消去（delete）セッション
  - 個別配布（send）セッション
  - タスク実行（execute）セッション
  - 個別収集（receive）セッション
  - 個別消去（waste）セッション
  - 連続実行（batch）セッション
  - 一括配布（deliver）セッション
  - 並行処理（race）セッション
  - 一括消去（clean）セッション
  - 処理全体（operate）セッション



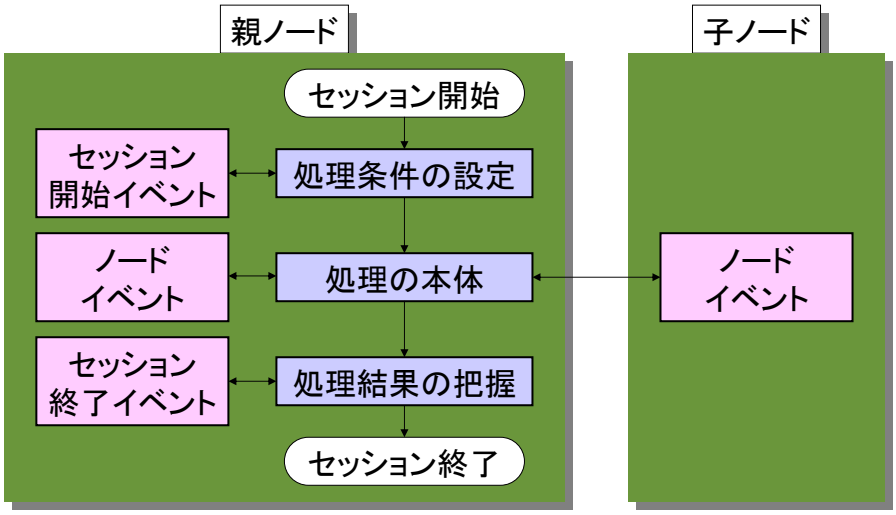
シーケンシャル実装とイベントドリブン実装

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP



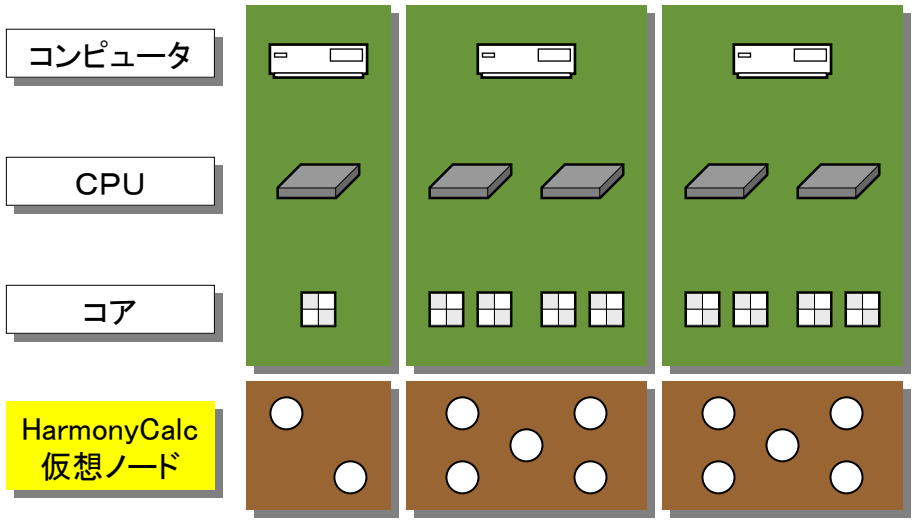
セッションイベントとノードイベント

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP



仮想ノード

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP



セッションモデルのメリット

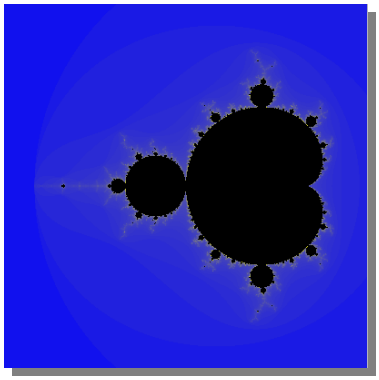
EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- 主なメリット
  - 並列化に伴うプログラムの変更が少ない
  - 並列化のために高度なプログラミング知識を必要としない
  - コンピュータの台数や、CPU・コアの個数に関係なく動作
  - 並列処理手順を変えても、ソースコードの可読性が失われない
  - 割り込み処理（処理完了率の取得、処理の中断）の実装が簡単
  - 並列プログラムの開発に「クラスタシミュレータ」が利用可能
- このような用途に最適
  - パッケージソフトウェアへのOEM組み込み
  - 対話型（コマンド型）アプリケーションの並列化
  - ライブラリの並列化
  - CやFortran以外で作られたプログラムの並列化

例題：マンデルブロ集合（1）

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

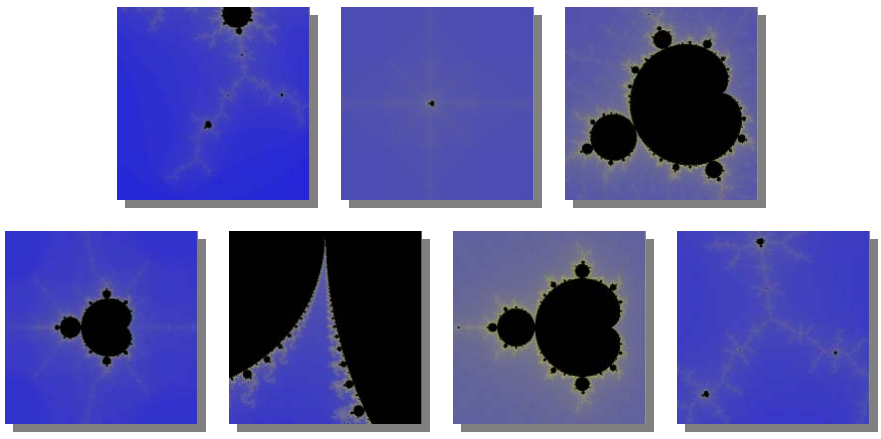
複素変換$Z \leftarrow Z^2 + C$を繰り返し、発散を判定
発散速度を色に変換し、複素平面にマッピング



例題：マンデルブロ集合（2）

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

拡大していくと、興味深い形が現れる



「マンデルブロ集合」プログラムの性質

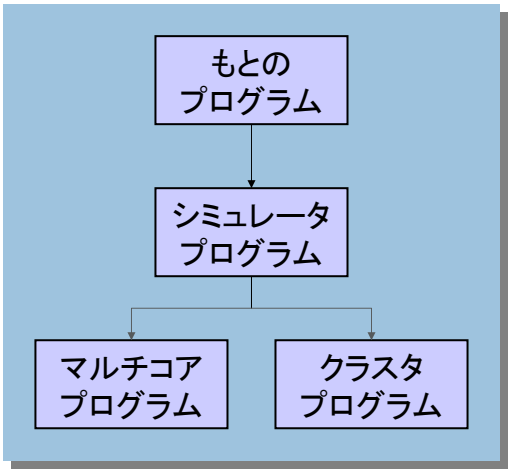
EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- 処理量が非常に多い
  - 160，000個のピクセルごとに、40，000回の複素計算が必要
  - 1枚の画像を得るのに、普及型パソコンで10分以上かかる
- 処理の順序は問わない
  - 画像をいくつかの小領域に分割して作ることができる
  - ピクセルを処理する順番を入れ替えても、結果は変わらない

並列化してみよう

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- シミュレータへ
  - DLLのリンク
  - イベントドリブン化
- マルチコアへ
  - DLLの交換
- クラスタへ
  - DLLの交換
  - イベントハンドラの切り分け



EMBARCADERO TECHNOLOGIES

DEVELOPER CAMP

おわりに

並列処理の二つの市場

EMBARCADERO TECHNOLOGIES

DEVELOPER CAMP

	科学・研究分野	産業・医療・民生分野
エンドユーザ	科学者・研究者、コンピュータ技術者	コンピュータや技術に詳しい人とは限らない
方向性・目的	高い処理性能を追求	限られた予算・人員で要求を充足
技術への要求	使い勝手よりも性能・自由度	性能・自由度よりも使い勝手
従来技術	パブリックドメインをベースに発展	未踏

始まりつつある「パラレルディバイド」

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

ハードウェアの並列化が進んでいくと、何が起こるか？

並列処理は面倒だから、取りあえず後回しにしよう	使えるものを使いながら、早く並列処理に慣れよう
↓	↓
ハードウェアを使いこなせない	ハードウェアを使いこなせる

並列処理は怖くない

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- 多くの人が、既存のテクノロジを使いこなすために膨大な努力を払い、多くの人が頓挫している
- しかし本当に重要なのは、対象のアルゴリズムを大きい粒度で分割するテクニック。テクノロジは後からゆっくり選べばよい
- 日ごろから並列処理を、「自分の技術」として意識していることが非常に大事。アルゴリズムの設計から変わってくる

ご清聴ありがとうございました

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

imageom
imageom

株式会社イマジオム

茨城県日立市水木町１－１１－１０
0294－28－0147
http://www.imageom.co.jp/
office@imageom.co.jp