システム評価法の確立
進捗状況の報告

2013/05/31

野村@東工大

目次

- 性能評価ツール
  - 性能取得ツールの整備状況
  - PAPIの妥当性検証
- 性能評価レポジトリ整備計画・状況
  - ORNLでの事例
- 性能評価セミナーのご案内
  - 6月最終週か7月頭を予定

アプリベンチマークの目的

- アプリの性能モデルを立てたときに、実際にそれが成り立っていることを確認する
  - 乖離している場合、どういう離れ方をしているか
    - モデルが不十分で考慮していないパラメータがある
    - 何らかの性能劣化要因がある
- 例: SCALE3ミニアプリのカーネル部分の性能モデル
  - FLOPS = システムのメモリバンド幅 / 実効B/F

目標関数　　カタログスペック　　実測 → How?

  - 例えばScalasca+PAPIでメモリアクセス(最外キャッシュミス)とFP演算を計測すれば測れる

性能取得ツール整備状況

- Vampir, Scalasca, Tau, PAPI
  - 京
    - I/O指標の取得(Tau)以外は動作
    - 各機能はログインノードか計算ノードのうち少なくとも片方で動く
    - パフォーマンスカウンタ(演算数etc)も正確
  - TSUBAME2.0
    - 一通りのツールは動作
    - パフォーマンスカウンタが一部不正確だが、workaroundは確立 (次ページ)

教訓: 得られた計測値を検証するミニベンチマークが必要

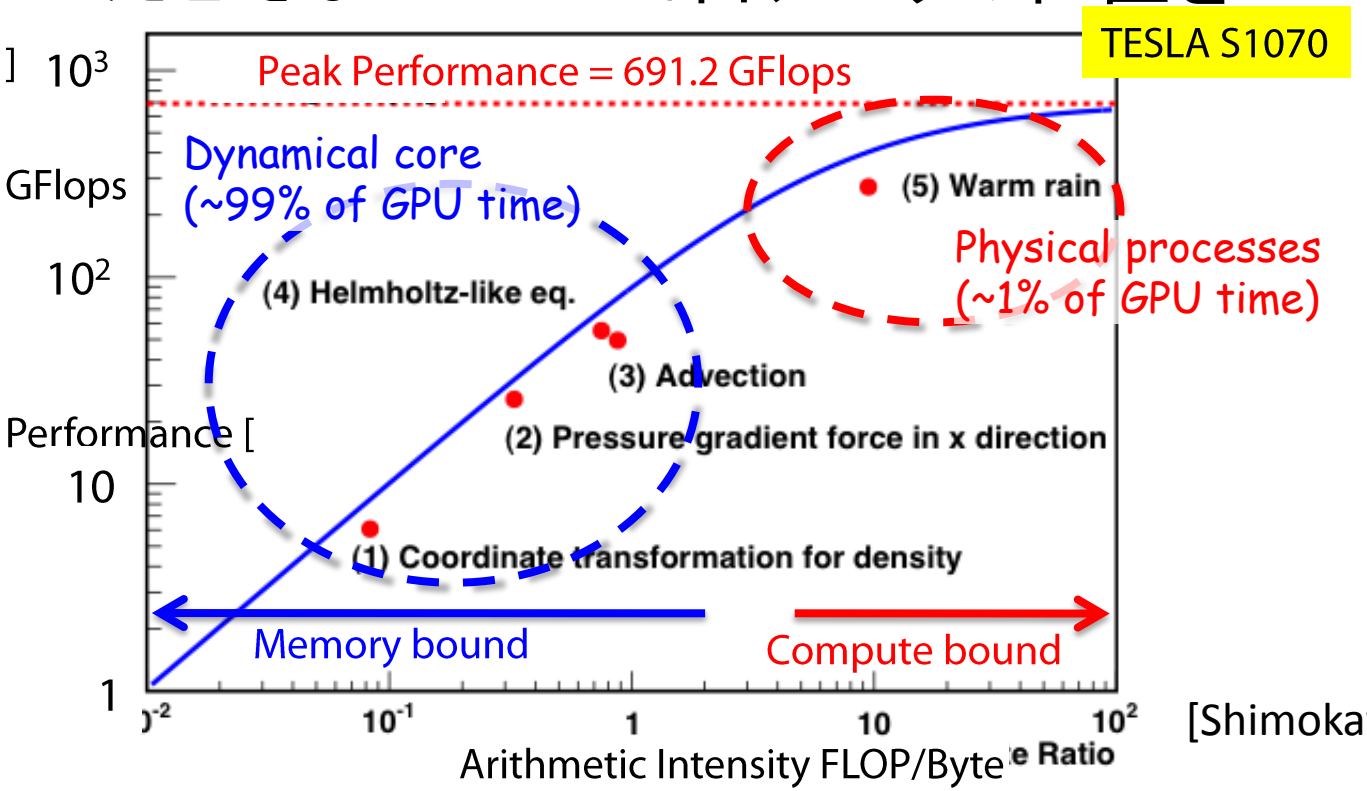
  - これらのツールの利用法講習会を予定

(参考) TSUBAME2.0(Westmere)のFPカウンタ

- Westmereのカウンタは以下の演算を別々にカウントしている
  - A: SP命令数(スカラー+SSE2)
  - B: DP命令数(スカラー+SSE2)
  - C: SSE2命令数 (SP4つもしくはDP2つに相当)
- PAPIはこれらを組み合わせて演算数を算出
  - SP演算数? = A + (4-1)*C
  - DP演算数? = B + (2-1)*C
  - カウンタCがSPとDPで重複してカウントされている
- Workaround
  - SPかDPか判断してCのカウントの行先を変える
    - 概算になってしまうが、SPもしくはDPに偏っているのであればこれで十分
  - SP演算数 = A + (A/(A+B)) * 3C
  - DP演算数 = B + (B/(A+B)) * C

性能評価リポジトリ

- Goal: ミニアプリでベンチマークを取って、Visualizeし、性能モデルを立てる
  - たとえばRooflineで自アプリの位置をプロット

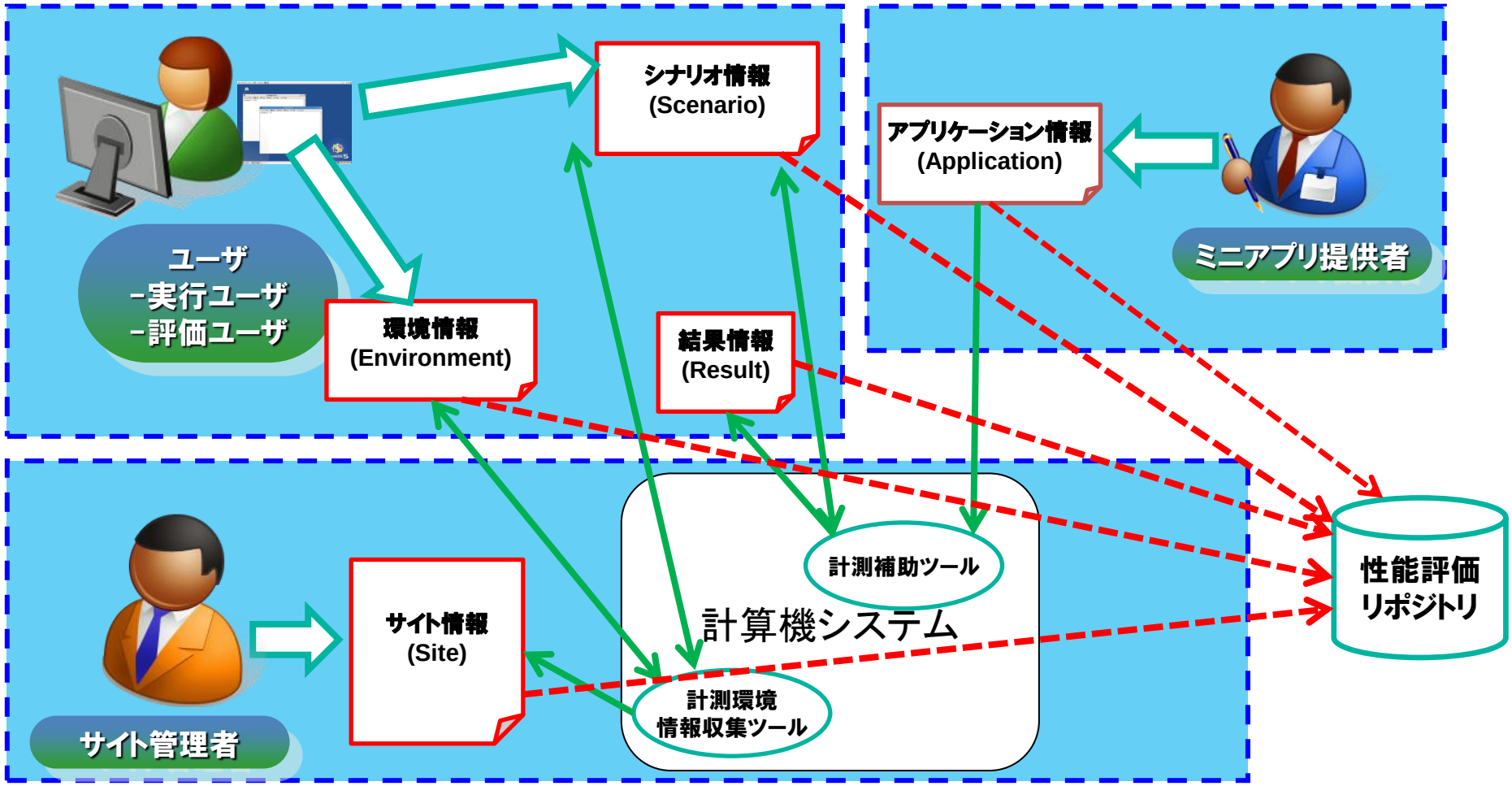


[Shimokawabe, et al]

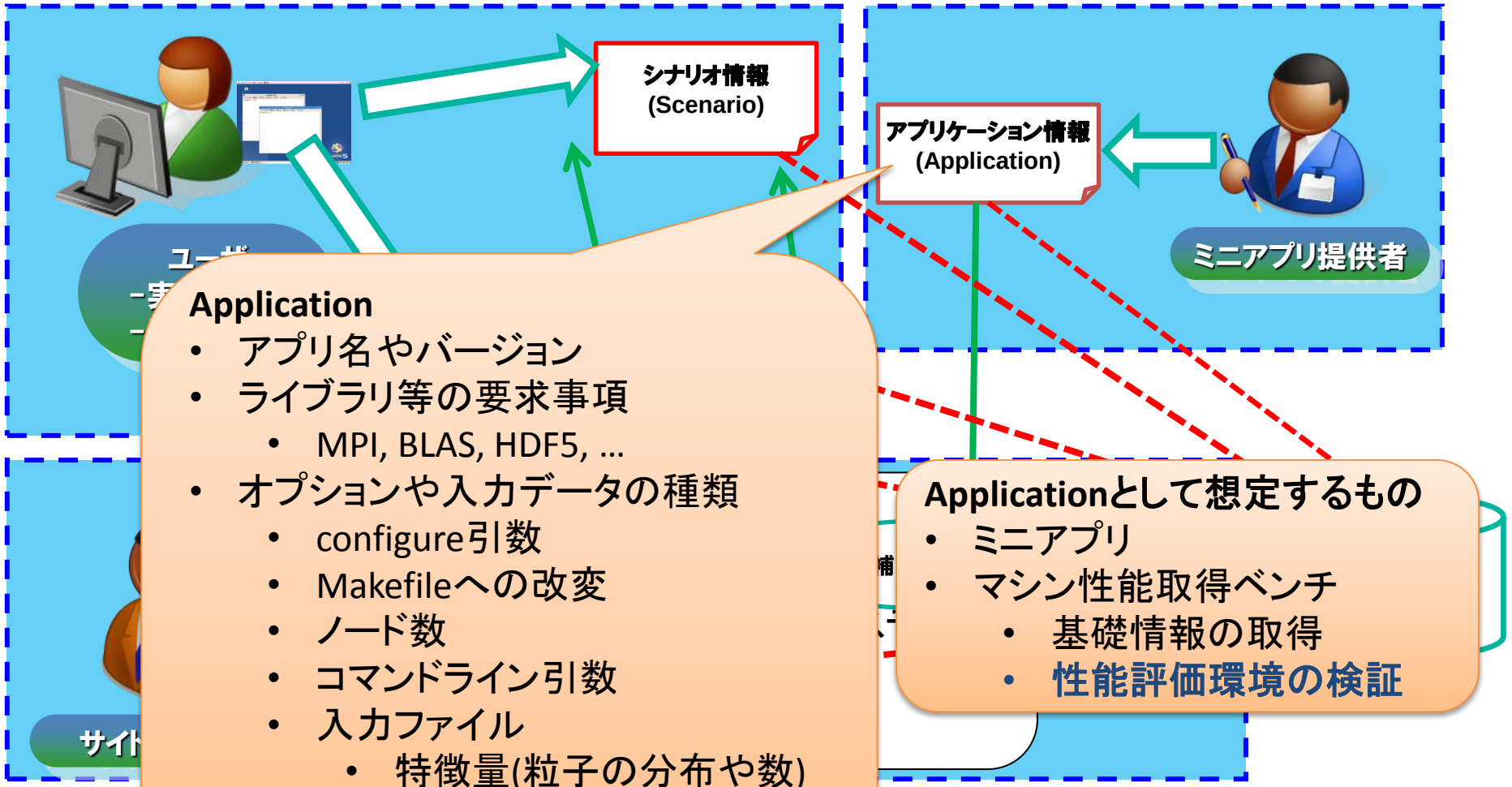
リポジトリで収集する情報

- 以下の5カテゴリのデータ・メタデータを収集
  - Application, Site, Environment, Scenario, Result
  - ベンチマークデータの表示・post mortem検証に必要十分なもの
  - 詳細は以降のスライドで説明
- それぞれを1つのXMLファイルにまとめる
  - これをコマンドラインツールやブラウザでUploadしていただく
  - 結果の表示はWebブラウザで行うことを想定

どの情報を誰が出すのか



どの情報を誰が出すのか



Applicationファイル

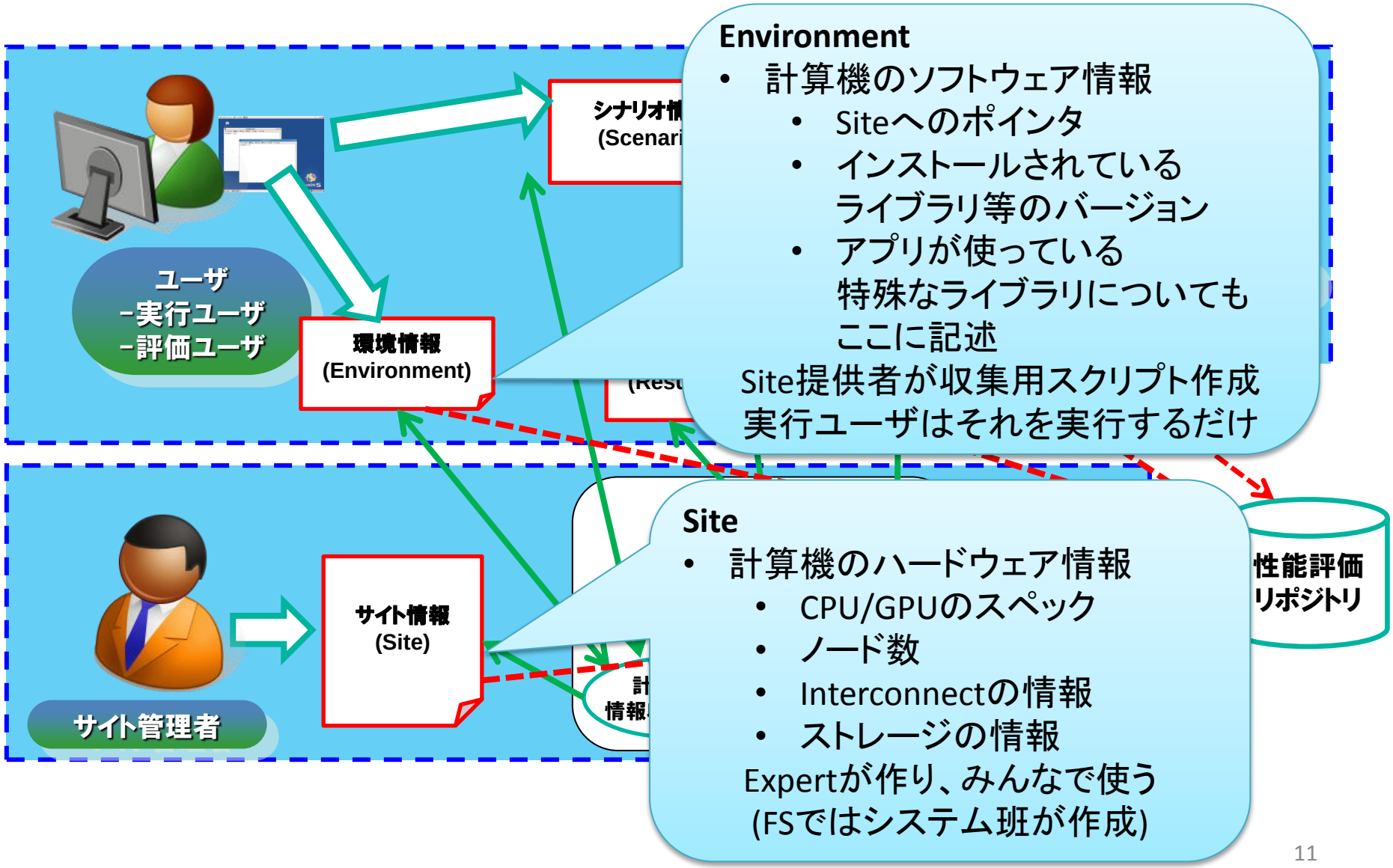
```
<application>
 <name>
 <version>
 <parameters>
  <parameter>
    <name>numprocs</name>
    <type>integer</>
  </parameter>
  <parameter> …
    <type>file</type>
  </parameter>
 </parameters>
 <configure>
  <option>--enable-XXX</option>
 </configure>
 <libraries>
   <require>mpi-c</>
   <require>lapack <version-gt>XX.X</version></>
 </>
 <execute>
  <option>-f <subst param="inputfile" /></>
 </execute>
 <regions>
   <region>
     <start>….f90:xxxx</start>
     <end>…</end>
   </region>
   <region>
    <procname>matmul</>
   </region>
 </regions>
</applications>
```

- 各ミニアプリの提供者が書く
- アプリの実行に関連する選択可能なオプションを網羅

- ミニアプリ制作時に必要なコンポーネントをここに明記する
  - ライブラリのバージョン指定とか

以下、XMLのスキーマは雰囲気の共有のための仮のもので今後確実に変更されます

どの情報を誰が出すのか



Siteファイル

```
<environment>
<name>TSUBAME 2.0 Thin</name>
<nodes>
<node>
  <processors>
    <processor>
      <name>Intel Xeon …</name>
      <arch type="x86_64">
        <feature>SSE2</feature>
        …
      </arch>
      <flops>…</flops>
      <numinstalled>…</…>
      …
    </processor>
    <processor>
      <name>Tesla M2050</name>
      <arch>nvidia_gpu</arch>
          <feature>sm_20</feature>
       …</arch></processor>
  </processors>
  <numinstalled> … </…>
</node>
</nodes>
<filesystems> … </…>
<interconnects> … </…>
</environment>
```

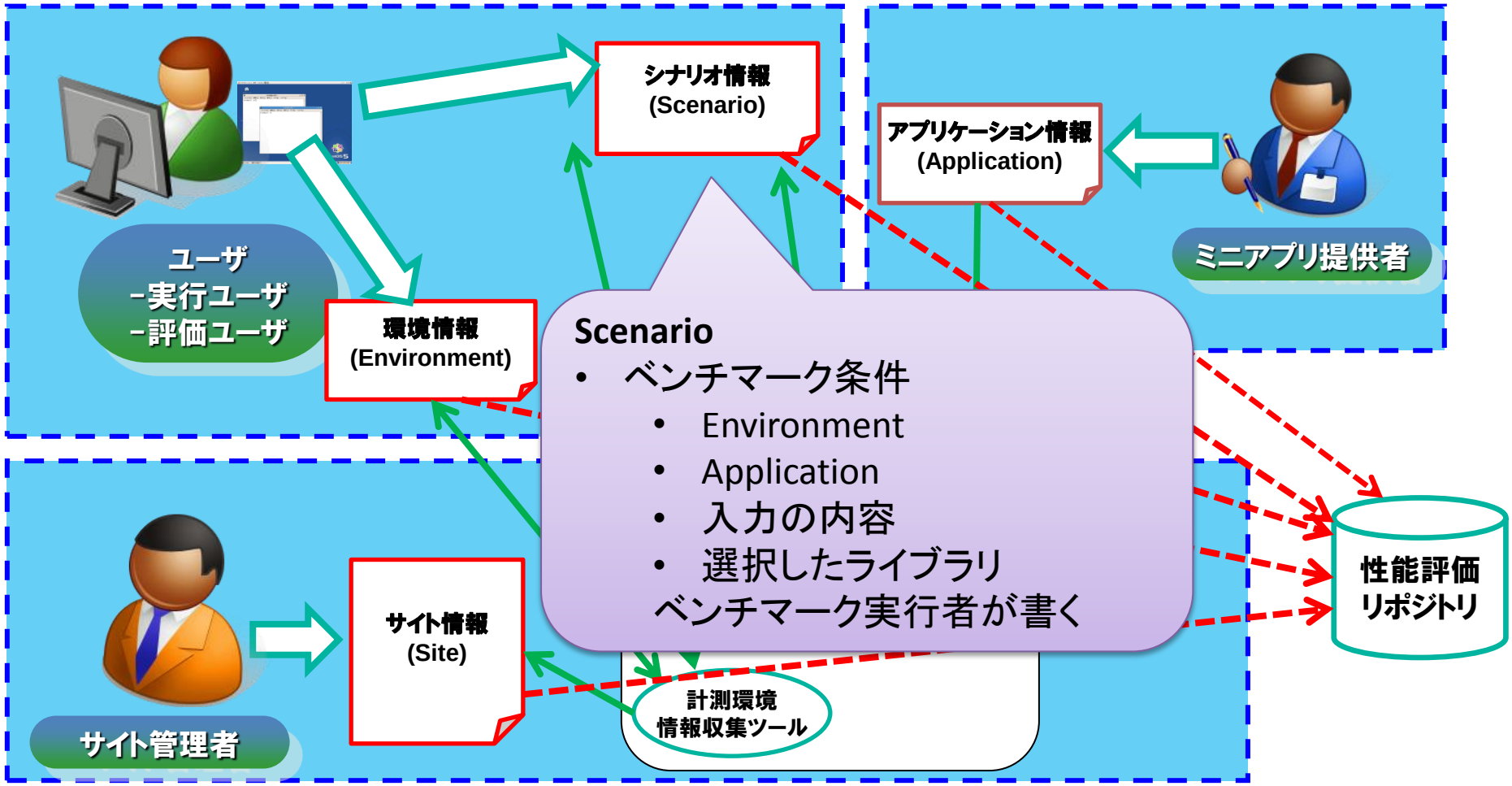
- 誰か(システム構築者)が書いて共有する
  - アプリFSシステム班の手の届く範囲はサンプルとして書く
- 基本的にはハードウェアの情報
  - 計測ごと、アプリごとで不変の情報のみ
  - 誰かが用意して、みんなで使う
- 計測環境情報収集ツールで簡易なテンプレートを出力するが、基本は手書き
  - 名前とかinterconnectsとか

Environmentファイル

```
<Environment>
 <site>http://…</site>
 <date>2013/05/31 00:00:00+0900</date>
 <compilers>
  <compiler name=“gcc-4”>
   <type>gcc</type>
   <version>4.0</version>
   <path>/usr/apps/…</path>
  </compiler>
  <compiler name =“icc”>…</>
 </compilers>
 <libraries>
   <library name=“papi-5.0.0”>
     <type>…</><version>…</>
     <path>…</path>
   </library>
 <libraries>
</senario>
```

- ある程度アシストしながら実行者に書いてもらう
  - Applicationファイルを見て必要な項目を出す
- システムソフトウェアの情報
- 各サイトごとにインストール規則に適合したスクリプト・テンプレートを書けば自動収集可能か
  - ls /usr/apps/free/ とかから入っているライブラリのバージョンを列挙したり
- ここに書くべきライブラリのminimum set はどこで定義される? → Applicationファイルで要求されているもの
  - コンパイラ
  - MPIライブラリ
  - BLAS
  - NetCDF等
  - Libc
  - Applicationファイルを読んで自動生成?

どの情報を誰が出すのか



Scenarioファイル

```
<scenario>
<environment>http://…</>
<application>http://…</>
<compiler>
  <type>gcc</>
  <version>4.0</>
</compiler>
<library>
  <type>…</><version>…</>
</library>
<option>
  <compilerflag>-DNODEBUG</>
  …
</option>
<parameter>
 <name>…</><value>…</>
</parameter>
<parameter>…</>
</scenario>
```

実行環境の曖昧性排除 (If needed)

アプリへの入力 (こっちがMain)

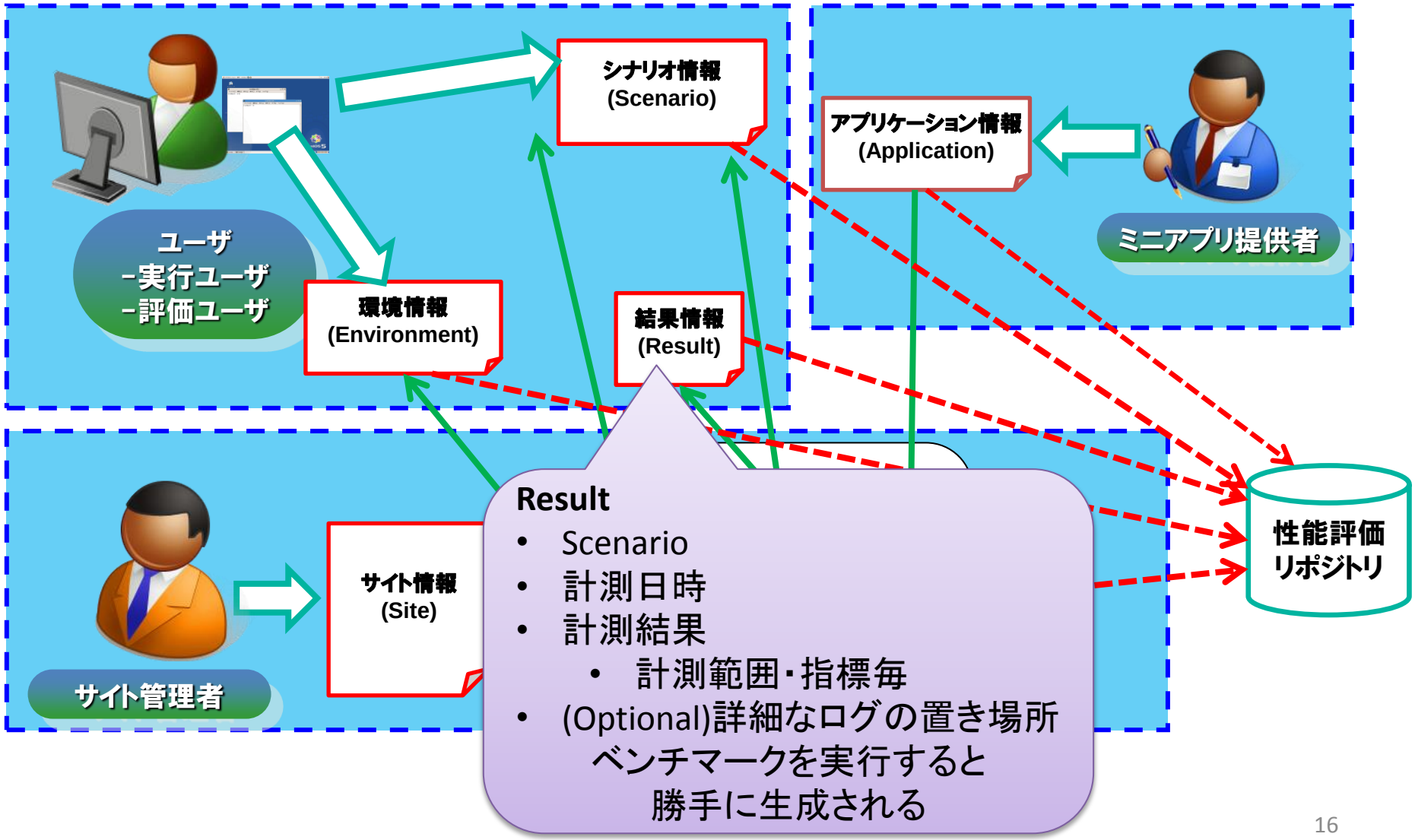
- 実行者が書く
- ここまでの情報をかき集めると、post mortem検証ができるデータが揃う
- 将来的に欲しい機能

Senarioファイルをパーズして、ライブラリ等があるかを確認する

パーズした結果をもとに実行に必要な以下のものを出力する

- PATHなどの環境変数
- ConfigureやMakefileへ与える項目
- ジョブスクリプトやqsubオプション

どの情報を誰が出すのか



Resultファイル

```
<result>
 <scenario>http://…</>
 <range>
  <name=“FFT”>
  <metric>
    <type>flop</>
    <value>2503413415</>
  </meric>
  <metric>
    <type>time</>
    <value>3.141592653</>
  </metric>
 </range>
 <range>
   …
 </range>
<result>
```

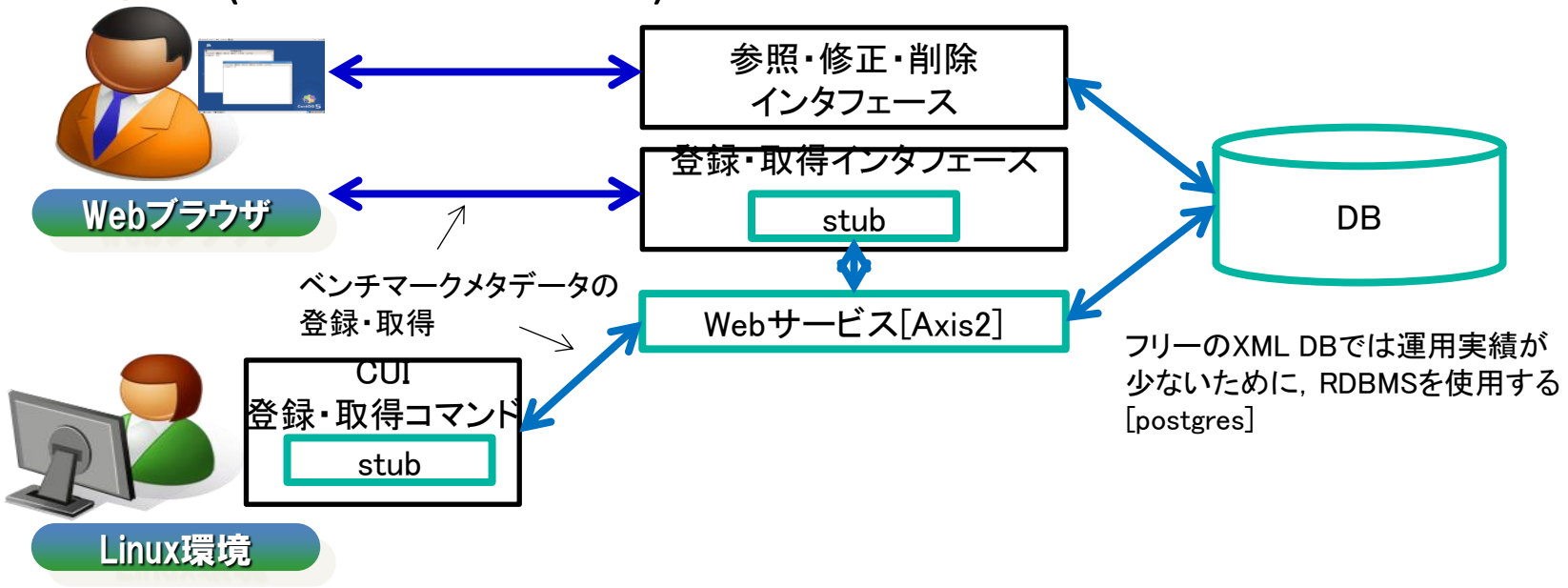
- ミニアプリ実行時にScenarioファイル等をもとに半自動生成
- ひたすら測定区間と計測結果の羅列
- ミニアプリを実行したら(半)自動で出力される
  - これをコマンドラインやwebでアップロードすればよい
  - カウンタを変えての複数回実行とか、データ纏めたりとか
- 何を測るのか(最低限これは取っておけリスト)というセットをどこで定義するか
  - 別途与える
  - これが入ってないファイルははねる
- アプリごとの追加計測項目はどこで定義するか → Applicationファイル
- (環境依存の)生データ(中間データ)の扱い
  - E.g. SL390のFlop値計測のためのカウンタ値
  - 解析には用いない
  - 想定される使い道
    - Resultファイルのデバッグ(あとから解析)用
    - 何か疑義が出てきたときのデータ置場
  - 基本的に保存する(URI Basedのポインタが欲しい)
    - 保存せずに済むように必要項目はここに埋める

ベンチマークで何を取得するか

- Minimum Set
  - FP演算数(精度毎), Int演算数
  - メモリアクセス量(R/W Bytes)
    - 各階層のキャッシュミス量
  - 通信量(Msg, Bytes, p2p相手毎およびcollective)
  - I/O量(Msg, Bytes, ファイルごと)
- Additional Set
  - メモリアクセスパターン
  - 電力消費
    - TSUBAMEは一部ノードで取得できるインフラがある

情報をどうやり取りするか

- それぞれXMLファイルを出力
  - 項目ごとに自動生成 or テンプレート作成
- これをSCM風のリポジトリへ投稿
  - コマンドライン or Web Interface
  - GitHubやSourceForgeのようなイメージ
  - 裏では可視化のためにDBへ保存
- 可視化(リスト・グラフ生成)はWebで



性能モデルの表現方法

- Aspen @ ORNL
  - アプリの本質的な挙動から必要な性能指標を定義し、実行時間等を見積もる
  - Aspenのパラメータとベンチマーク結果を結びつけることで、モデルと性能を比較できるようにする予定

```
model XXX {
  param XXX = …;    ← 入力や一部マシンパラメタ
  …}
kernel YYY {
  exposes parallelism [n^2]
  requires flops […] as dp, simd
…}
control ZZZ {
  YYY
  AAA -> iterate [5] { BBB } }
```

ORNL(DOE)での動き

- 日本のアプリFSと同様にミニアプリ(proxy-app)をもとにexa-scaleのマシンの性能評価を行う研究が始まっている
  - 我々と同レベルの進捗度、類似点もかなり多い
  - Proxy-appは実アプリのカーネルごとに切り出す
    - 元アプリの収集数もほぼ同じ (20本弱)
    - Proxy-appはオープンソースにする予定
  - 性能評価ツールは内製が多い
    - 我々は既存のツールの活用がメイン
- Aspenの利用を含め、共通するところは共同で開発する予定
  - 究極的にはUS, Europeのマシン・アプリを評価に使えるようになる

性能評価に関するセミナー

- 京およびTSUBAME2.0を対象とした性能評価ツールに関するセミナーを開催予定
  - Scalasca, Tau, VampirTraceを想定
    - デプロイ方法と利用法
  - 対象: 京アカウント保持者(4FS関係者)
  - 時期: 6月最終週～7月初旬
    - 都合の良い時期についてお聞かせください
  - 会場: 東工大 + AICS (Polycomで接続予定)

今後の東工大グループの予定

- フルアプリおよびミニアプリの性能測定
  - 性能評価ツールの妥当性検証に手間取っていたため、まだ実行できていない
  - 6月～7月前半で集中的に測定する予定
    - マシン繁忙期および節電要請期を避けたい
- 性能評価リポジトリおよび周辺ツール開発
  - データ形式のスキーマ確定
  - 可視化・Aspenとの繋ぎこみ
  - 周辺ツール・インフラの構築