5G-2

共通指標を用いた複数仮想マシン動作時における性能見積もり方式

藤若　雅也 †　　竹村　俊徳 †

† 日本電気株式会社　サービスプラットフォーム研究所

1　はじめに

仮想化技術の普及、物理サーバの性能向上に伴い、単一の物理サーバ上で複数の仮想マシンが実行されるのは一般的になりつつある。多くの仮想マシンを単一の物理サーバ上に集約させることで運用コストを押さえることが可能だが、複数の仮想マシンを実行することによる性能劣化の問題が報告されている。複数の仮想マシン実行時の性能を把握することは、顧客との SLA の維持しつつ、運用コストの削減する上で重要である。

2　提案システム

我々は複数の仮想マシン実行時の性能を見積もるためのオフライン型プロファイリングシステムを実装した。本システムは、物理サーバへのプロファイリング用仮想マシンの配備、仮想マシン上でのワークロードの生成、性能計測および性能のモデル化を自動的に行う。

2.1　性能モデル

仮想マシンの性能モデルは、ワークロード生成時のリソース使用量を計測し、統計手法を用いてモデル化する。リソースは、CPU、ネットワーク I/O、ディスク I/O などである。ワークロードの選択には自由度があるが、我々は単一のリソースの消費に特化した複数種のワークロードを用いた。

本稿では、ワークロードの種別数を m としたとき、各ワークロードを $w_i(1 \le i \le m)$ と記し、リソース β の使用量を L_β と記す。また、ある単一のワークロード w_i を x だけ発生させたときのリソース β の使用量を $L_\beta^{w_i}(x)$ とし、複数種のワークロード $\boldsymbol{W} = (w_1, w_2, \cdots, w_m)$ を $\boldsymbol{X} = (x_1, x_2, \cdots, x_m)$ だけ同時に発生したときのリソース β の使用量を $L_\beta^{\boldsymbol{W}}(\boldsymbol{X})$ と記す。

2.2　ワークロードの生成方式

ワークロード w_k の仮想マシン上で実行可能な最大値を w_k_max とおく。仮想マシン上で発生させるワークロードの値は、分割数 N としたとき、ワークロードの 0〜w_k_max を N 分割した $N+1$ 通りの離散値をサンプリングポイントとする。m 種のワークロードを同時に発生させる場合、各ワークロードの組合せ（$(N+1)^m$ 通り）をサンプリングポイントとする。

分割数 N を大きく設定し、より多くの点でサンプリングすることで、精度の高いモデルを構築することが可能となるが、当然プロファイリング時間の増加を招く。特に複数種のワークロードの組合せ発生を行う場合、組合せ数がワークロード種別数 m のべき乗で効いてくるため、膨大の時間を要する。例えば、ワークロード種別数 m を 5、分割数 N を 10、サンプリング実行時間を 60 秒とした場合、プロファイリングに 2600 時間以上の時間を要してしまう。

2.3　単一仮想マシン実行時の性能モデル化

モデル化の対象のリソースを β とする。単一仮想マシン実行時の性能モデル化では、仮想マシン上で複数種 (m 個) のワークロード $\boldsymbol{W}$ を発生させた際のリソース β の使用量を $L_\beta^{\boldsymbol{W}}(\boldsymbol{X})$ をモデル化する。

単純なモデル化の手法としては、

a. $(N+1)^m$ 通りのサンプリングを行い、計測した性能データから $L_\beta^{\boldsymbol{W}}(\boldsymbol{X})$ を直接モデル化する
b. ワークロード w_k の値 x_k を単独で発生した際のリソース β の使用量 $L_\beta^{w_k}(x_k)$ をワークロードごとにモデル化し、各モデルの線形結合で $L_\beta^{\boldsymbol{W}}(\boldsymbol{X})$ を近似する

が挙げられるが、a の手法は前述したようにプロファイリングに膨大な時間がかる問題がある。一方で、b はワークロード種別毎にモデル化をするため、複数種のワークロードを同時に実行した際の振る舞いをモデル化することができない。

そこで我々は a,b を組み合わせた手法を用いて、上記の課題を解決するプロファイリング方式を提案する。

2.3.1　プロファイリングアルゴリズム

k 種類のワークロードに対する性能モデルを k 次性能モデルと呼ぶこととする。プロファイリングは以下のステップで行う。

1. 1 次性能モデルの構築
   2.3-b の前半部分に相当し、単一ワークロード w_k に対するリソース β の性能モデル $L_\beta^{w_k}(x_k)$ を構築する。1 次性能モデルは N_1 個のワークロードを発生させ、計測した性能データを 4 次の多項式
   $$L_\beta^{w_k}(x_k) = a_0 + \sum_{i=1}^{4} a_i \cdot x_k^i \qquad (1)$$
   に当てはめることによってモデル化する。また、1 次性能モデルは、ワークロードの数だけ、つまり m 個作られる。
2. 2 次近似性能モデルの構築
   2.3-b の後半部分に相当し、m 個の 1 次性能モデルの 2 つを組み合わせ、近似性能モデルを構築する。つまり、
   $$\widehat{L}_\beta^{(w_i,w_j)}(x_i, x_j) = g(L_\beta^{w_i}(x_i) + L_\beta^{w_j}(x_j)) \qquad (2)$$
   によって構築する。$g(.)$ は最大性能値を超えないように上限を与える関数である。2 次近似性能モデルは ${}_mC_2$ 個構築される。

A performance estimation method using common index for multiple virtual machine environments

3. 2 次近似性能モデルの補正
   2 種類のワークロードを同時に発生させ (分割数 N_2)、得られた性能データと 2 次近似性能モデルの推定値の誤差を求め、誤差データをスプライン関数を用いてモデル化したものを $D_\beta^{(w_i,w_j)}(x_i,x_j)$ と置く。$D_\beta^{(w_i,w_j)}(x_i,x_j)$ を用いて

$$L_\beta^{(w_i,w_j)}(x_i,x_j) = \widehat{L}_\beta^{(w_i,w_j)}(x_i,x_j) + D_\beta^{(w_i,w_j)}(x_i,x_j) \quad (3)$$

   のように補正したものを 2 次性能モデル $L_\beta^{(w_i,w_j)}(x_i,x_j)$ とする。
4. k 次近似性能モデルの構築
   k 次近似性能モデルは $k-1$ 次性能モデルから以下のように近似する。

$$\widehat{L}_\beta^{\boldsymbol{W}^k}(\boldsymbol{X}^k) = \mathrm{worst}_{i\le k}(L_\beta^{\boldsymbol{W}_i^{k-1}}(\boldsymbol{X}_i^{k-1}) + L_\beta^{w_i}(x_i)) \quad (4)$$

   ただし、$\boldsymbol{W}^k = (w_1,\cdots,w_k)$、$\boldsymbol{X}^k = (x_1,\cdots,x_k)$ とし、$\boldsymbol{W}_i^{k-1}$ は $\boldsymbol{W}^k$ のうち i 番目の要素の除いたものを指す。$\boldsymbol{X}_i^{k-1}$ も同様。worst は $i \le k$ のうち、性能が最も悪い値を返す関数である。
5. k 次近似性能モデルの補正
   補正は 2 次の場合と同様に行う (分割数 N_k)。
6. $k = m$ になるまで上記プロセスを繰り返す

各近似性能モデルの補正段階で、複数のワークロードの組合せを実行するが、その際の分割数を高次になるに従い小さく設定する（つまり、$N_i > N_j (\text{if } i < j)$）ことで全体のプロファイリング時間を削減することが可能となる。

2.4 複数仮想マシン実行時の性能モデル化

複数仮想マシン実行時の性能モデル化では、既に複数の仮想マシン (N_{vm} 台とする) が動作しているところに、ある仮想マシン VM^p を追加した際の性能 $\widetilde{L}_\beta^{\boldsymbol{W}^m}(\boldsymbol{X}^m)$ を求めることを目的とする。~は複数の仮想マシンの競合を考慮していることを意味し、前述の $L_\beta^{\boldsymbol{W}^m}(\boldsymbol{X}^m)$ とは異なる。便宜上 VM^p の $\widetilde{L}_\beta^{\boldsymbol{W}^m}(\boldsymbol{X}^m)$ を $\widetilde{L}_\beta^{VM^p}(\boldsymbol{X}^m)$、$L_\beta^{\boldsymbol{W}^m}(\boldsymbol{X}^m)$ を $L_\beta^{VM^p}(\boldsymbol{X}^m)$ と記す。また、既に動作している仮想マシンのリソース β の使用量を $L_\beta^{VM^i}$ とおき、その総和を $L_\beta^{VM^{all}}$ とする。仮想マシンに対し、均等なスケジューリングがされると仮定すると

$$\widetilde{L}_\beta^{VM^p}(\boldsymbol{X}^m) = \quad (5)$$

$$\begin{cases} L_\beta^{VM^p}(\boldsymbol{X}^m) & \text{if c1} \\ \max(l_\beta\_max - L_\beta^{VM^{all}},\ l_\beta\_\max/(N_{vm}+1)) & \text{elseif c2} \\ L_\beta^{VM^p}(\boldsymbol{X}^m) & \text{otherwise} \end{cases}$$

と近似することができる。ただし
条件 c1 は $L_\beta^{VM^p}(\boldsymbol{X}^m) \le l_\beta\_max/(N_{vm}+1)$、
条件 c2 は $L_\beta^{VM^p}(\boldsymbol{X}^m) + L_\beta^{VM^{all}} \ge l_\beta\_max$
である。l_β_max はリソース β の最大性能値を指す。

3 評価と考察

3 種のワークロード (CPU、ネットワーク送信、ネットワーク受信) を用いて単一仮想マシンの性能モデルを構築した。モデル化の対象リソースは CPU(使用率) とした。評価サーバは Express5800/Xeon3.60GHz(2core)、仮想ミドルは Xen を用いた。評価では以下の 3 つの手法の比較を行った。

- 従来手法: 2.3-a に記述した 3 次性能モデルを直接求める方法 (計 55 サンプリング)
- 提案手法 (1 次 30, 2 次 15, 3 次 10 の計 55 サンプリング)
- 提案手法 (変形版): 2.3-b を用いて 3 次近似性能モデルを直接構築し (2 次の工程を省略)、補正により 3 次性能モデルを求めた場合 (1 次 30, 3 次 25 の計 55 サンプリング)

モデル化におけるサンプリング数は全て同一になるように設定した。() 内はモデル構築に用いたデータ数を示す。また、モデルの評価用データとしては、1331(各ワークロードの分割数を 10 とした場合の総組合せ) のサンプリングデータを用いた。

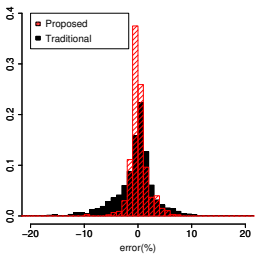

図 1

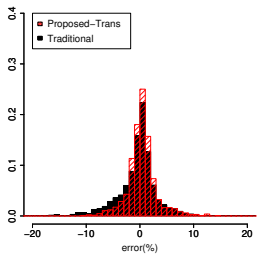

図 2

図 1 は、提案手法と従来手法における評価用データのフィッティング誤差をヒストグラム表示したものである。従来手法に比べ、提案手法は精度よくモデル化できていることがわかる。また、図 2 は提案手法 (変形版) と従来手法の比較を行ったものである。こちらは、従来手法に比べあまり精度は向上していない。これは 2 次の性能モデル化が精度向上に重要であることを示している。

4 今後の展望

今後はディスク IO などを含めた複雑なケースについて本提案方式の効果を検証する予定である。

5 謝辞

本研究の一部は、総務省委託研究「クラウドサービスを支える高信頼・省電力ネットワーク制御技術の研究開発」の成果である。

参考文献

[1] Stefan Appel, Ilia Petrov, Alejandro Buchmann: PERFORMANCE EVALUATION OF MULTI MACHINE VIRTUAL ENVIRONMENTS, In SPEC Benchmark Workshop, 2010.
[2] Tickoo Omesh, Iyer Ravi, Illikkal Ramesh, Newell Don: Modeling virtual machine performance: challenges and approaches,SIGMETRICS Perform 2009.