特集論文

大規模IoTシステムにおける計算機リソースサイジング技術

魚住光成* 鶴　薫*
小杉　優*
佐藤尚也*

Computer Resource Sizing Technology in Large-Scale IoT Systems

Mitsunari Uozumi, Yu Kosugi, Naoya Sato, Kaoru Tsuru

要　旨

近年の計算機とネットワークの技術の発展と普及によって，様々なモノの間で情報交換や制御を行うことが可能となってきた。IoT(Internet of Things)システムはセンシングしたデータに対して何らかの処置を施して記憶し，照会に対して記憶からデータを引き出して応えるようなデータ処理を行う。

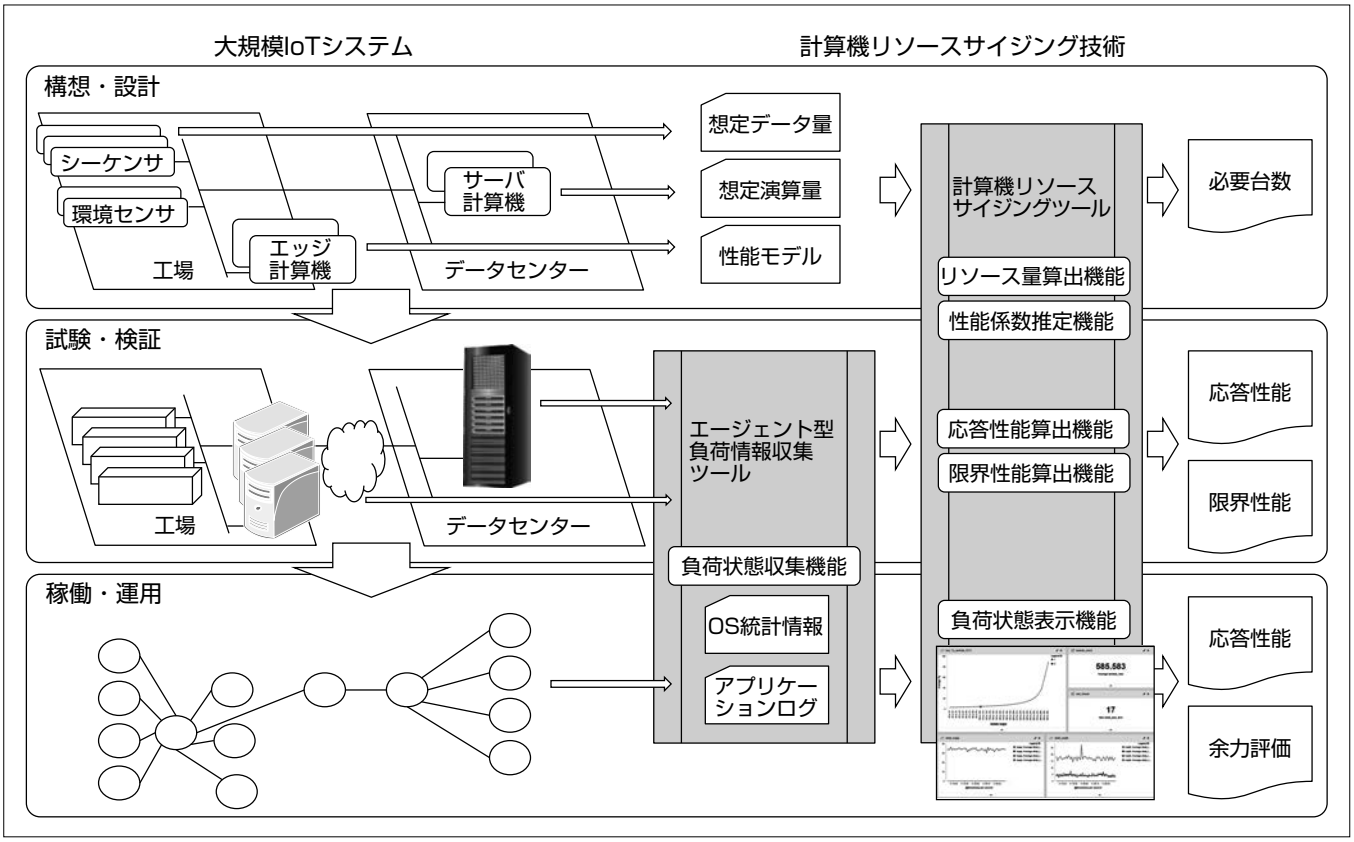
計算機とネットワークの選択の幅が広がったことで，IoTシステムは様々なハードウェア構成を取り得る。しかし，目的の処理を実現するシステム構成の構成要素の処理能力，数量の確定が，システムの構築で課題となる。

三菱電機の計算機リソースサイジング技術は，システムの構想段階から検証，運用開始後まで，計算機リソースの使用状況を予測，計測，評価し，システムの安定した構築と稼働を可能とする。

IoTを可能とした計算機とネットワークの技術の発展は，システムの複雑化ももたらしており，直感的なサイジングは不可能である。

当社の計算機リソースサイジング技術では，システムの構成要素を還元的に分解し，それぞれのトラフィックと処理能力を評価することで，ボトルネックとなる要素を抽出するとともに，パッケージソフトウェアなどブラックボックス化された構成要素は，その性能特性をモデル化することで，システム全体の性能の予測，評価を行う。

この技術によって，形態が多様化した大規模IoTシステムの計算機リソースのサイジングを可能としている。



大規模IoTシステムと計算機リソースサイジング

大規模IoTシステムでは，数多くのデータ発生源を対象とするため，計算機リソースのサイジングは開発ステップごとに繰り返すことが重要である。当社の計算機リソースサイジング技術は構想段階から運用監視後までの開発ステップに対応したツールを用意し，システムの開発者，運用者が計算機リソースの必要台数の算出から稼働後の処理能力の余力の評価まで行うことを可能としている。

*情報技術総合研究所

1. まえがき

近年の計算機とネットワークの技術の発展と普及によって，様々なモノの間で情報交換や制御を行うことが可能となってきた。

計算機は事業者が設備を抱えるオンプレミスといわれる形態から，計算機リソースをサービスとして提供するクラウドサービスの利用，エンドユーザーに近いところで演算を行うエッジコンピューティングを可能とする安価な計算機リソースまで，選択の幅は広がった。ネットワークは，インターネットや専用線によるプライベートネットワーク，論理的にプライベートなネットワークを実現する各種VPN(Vertual Private Network)などが，有線，無線のネットワーク技術に支えられて実現している。

これらを背景に，個々のモノが相互にネットワークを介して結び付くIoTが現実のものとなってきている。

2. 大規模システムのサイジング

大規模システムの計算機リソースのサイジングは古くから行われている[1][2][3][4][5][6]。

1980年代，大規模な情報処理システムは計算機センターなどに設置された計算機を利用して構築された。当時の情報処理システムは，リソースの割り付けや実行モジュールなどを定義した処理単位をジョブとして記述し，計算機のキューに投入するバッチジョブ型，多数の端末からネットワークを通じて計算機をアクセスするオンライン型が主流であった。

ここで行われたサイジングと大規模IoTシステムを対象としたサイジングの基本的な考え方は変わらない。

2.1 バッチジョブ型

利用者はJCL(Job Control Language)などで記述したジョブを入力装置から投入する。JCLにはアプリケーションプログラムのロードモジュールからアプリケーションプログラムが入力するファイル，出力ファイルや出力装置などが記述されており，ジョブとして一連の処理が完了するまで，利用者が介在することはない。投入されたジョブはジョブキューに蓄えられ，計算機は1件ずつジョブを取り出し演算装置でロードモジュールを実行する。

利用者から見ると，ジョブを投入してから結果を出力装置から取り出すまでに数時間を要する場合もある。一方，演算装置はジョブキューにジョブがたまっている間はその演算能力の100%が使用される。厳密にはディスクアクセスなどCPUと比較して遅いデバイスへのアクセスも伴うため，CPUがアイドルな時間も発生する。これを回避する範囲で複数のジョブを多重に実行することもあるが，今度はディスクアクセスの性能に拘束される。

したがって，ジョブの平均実行時間がTsのとき，ジョブの投入が1/Ts(件/秒)以上行われるとジョブキューは無限大に長くなる。投入したジョブが完了するまでの時間に運用上の上限の設定がないならば，1/Tsがバッチジョブ型の限界性能である。これを超えるためにはより速い演算装置を用意する必要がある。

2.2 オンライン型

利用者が端末を操作すると，計算機の端末制御装置は入力されたデータを受け取り，演算装置上で動作しているオンラインアプリケーションに渡す。オンラインアプリケーションはデータに対する処理を実行して，その結果を，端末制御装置を介して端末に返す。複数の端末がある場合，端末制御装置は端末から受け取ったデータを逐次オンラインアプリケーションに渡し，オンラインアプリケーションはデータを順番に処理していく。

このデータ処理にかかる平均実行時間をTsとするとき，端末からの1/Ts(件/秒)のデータ到着が限界性能であるが，実際は端末に対する応答性能を確保する必要があるため，実用上の限界性能はこれを下回る。

データがランダムに到着する場合，0.5/Ts(件/秒)のとき平均応答時間はTsの2倍となり，この負荷を超えると応答性能は急速に劣化し，1/Tsのとき無限大となる。

3. IoTシステムのサイジングの課題

IoTを可能とした計算機とネットワークの技術の発展は，システムの複雑化ももたらしており，直感的なサイジングは不可能なだけでなく，古典的なサイジングの手法も，そのままでは適用が難しくなっている。

幾つかの課題を解決しないと，IoTシステムの計算機リソースのサイジングはできない。

3.1 用途に応じた様々な形態

IoTシステムは，その用途に応じて様々な形態をとる。形態が異なると構成要素間のトラフィックも変わるため，IoTシステム用のサイジングといった一律の手法は成立しない。

3.2 ブラックボックスの増加

システムを構成する要素の多くが，第三者が開発したパッケージソフトウェアなどであるため，その性能特性が不明であることが多い。

3.3 プログラム構成の複雑化とマルチコア

1台の計算機が複数のCPUコアを持つことは一般的になってきている。これを有効に利用するため，ソフトウェアも複数のCPUの割り付け単位を同時に持つマルチスレッドの実装形態を取ることが多い。

こうしたマルチスレッド，マルチコアの構成では，論理的に処理要求を受け付けるスレッドと，実際に演算を行うCPUコアの2層構造となるため，複数の要求を同時並行に処理した場合の時間性能の予測が難しくなっている。

3.4 計測作業の長時間化

データ発生源が多く，また複数の計算機を使用するため，

試験・検証での計測作業が長時間化している。多くの場合，時間性能の評価は計測の事後に行われるため，システムの設定や計測の誤りがあると，もう一度最初から計測作業を行わなければならない。

4. IoTに対応したサイジング技術

当社の計算機リソースサイジング技術では，システムの構成要素の還元的な分解，ブラックボックス化された構成要素の性能特性のモデル化，マルチスレッドに対応した応答性能算出，結果が即時に確認できる負荷情報収集と可視化を実現することで，大規模IoTシステムにおける計算機リソースサイジングを可能にした。

4.1 システム構成要素の還元的分解

計算機リソースサイジング技術では，システムの構成要素を還元的に分解し，それぞれのトラフィックと処理能力を評価することで，ボトルネックとなる要素の抽出とサイジングを行う方法をとる。

IoTシステムはセンシングしたデータに対して何らかの処置を施して記憶し，任意のタイミングで発生する照会に対して記憶からデータを探索して応えるようなデータ処理を行う。このデータ処理のモデルを**図１**に示す。システムの設計で，図に示した要素は何台かの計算機に分割されて実現される。

例えば，一極集中型の実現方式では**図２**のように，センシングと照会を行う計算機，処理と記憶を行うサーバ計算機の構成になる。この場合，サイジングを行うべき計算機はサーバ計算機１台となり，入力のトラフィックと処理の実行時間を計算機リソースサイジングツールで評価すれば，サーバ計算機として必要な処理能力や台数は決定できる。計算機リソースサイジングツールでは，この算出を待ち行列モデルに基づいて行っている。

センシングの近傍にエッジ計算機を置き，処理をサーバ計算機と分担しあうエッジコンピューティング型の構成を**図３**に示す。エッジ計算機がデータの処理をサーバ計算機と分担することで，サーバ計算機の負荷を軽減するとともに，データの集約によるサーバ計算機が受けるデータ数の削減が行える。エッジ計算機とサーバ計算機それぞれの入力のトラフィックと処理の実行時間を計算機リソースサイジングツールで評価すれば，エッジ計算機とサーバ計算機に必要な処理能力や台数は決定できる。

このように，構成要素に還元して計算機リソースサイジングツールでサイジングを行えば，システム全体で必要な計算機の全体の規模を導き出すことができる。

4.2 性能特性のモデル化

計算機リソースサイジング技術では，パッケージソフトウェアなどブラックボックス化された構成要素は，その性能特性をモデル化することで性能評価に組み入れている。

計算機リソースサイジングツールは，計測したデータから対象の性能特性を式の係数として出力する。この式は，以下のような関数としている。

$$T_q = \mathrm{mmn}(n,\ \lambda,\ T_s) \quad \cdots\cdots (1)$$

ブラックボックスであっても多重度n，単体処理時間T_sを係数として導出することで，任意のトラフィックλでの応答時間T_qを算出することが可能となっている。

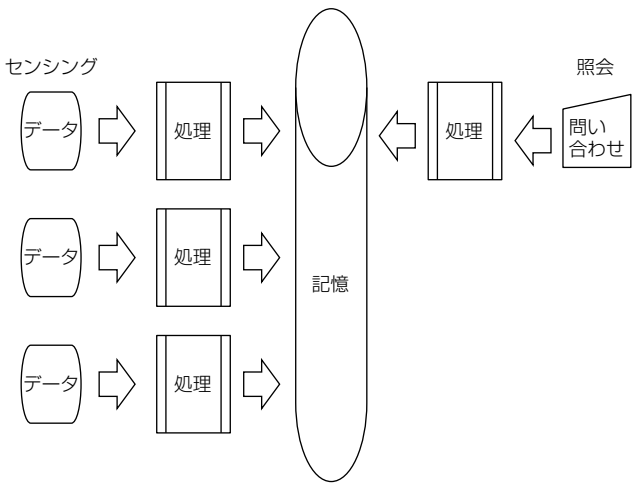


図１．IoTシステムのデータ処理

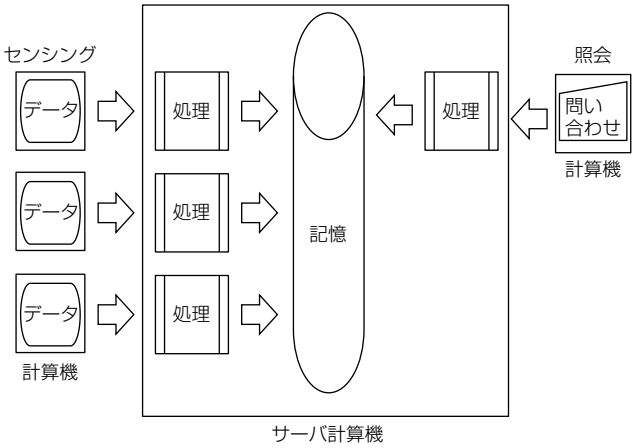


図２．一極集中型のシステム構成

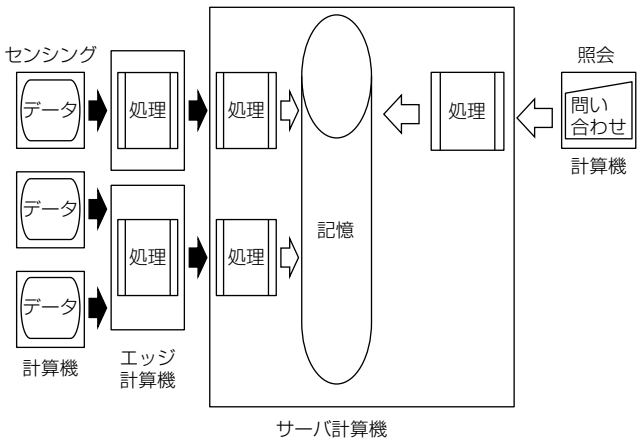


図３．エッジコンピューティング型のシステム構成

4.3　マルチスレッドに対応した応答性能算出

計算機リソースサイジングツールでは，複数のスレッドがマルチコア上で動作する場合の性能を，**図4**に示すような多段の待ち行列で算出している。

アプリケーションはトラフィックλをn個のスレッドで処理しており，それぞれのスレッドはCPU要求を出す。このCPU要求はm個のコアの内，空いているCPUコアに割り付けられる。この簡略化したモデルによって，任意のトラフィックλにおける応答時間T_qの算出を可能としている。

4.4　負荷情報収集と可視化

計算機リソースサイジングツールは，エージェント型負荷情報収集ツールと連携することで，計測中の性能評価結果の表示を実現している。

図5に性能評価結果表示の画面例を示す。図の左上は，直近の計測値を点で，トラフィックが変わった場合の応答時間の予測値を線で示している。右上の2つの数値は，アプリケーションの管理指標を示している。下段は，計測対象のリソース消費量の時系列な推移を示している。

この機能によって，現在行っている計測が妥当なものか即時に判断できるので，手戻りによる計測作業の効率低下を抑止できる。

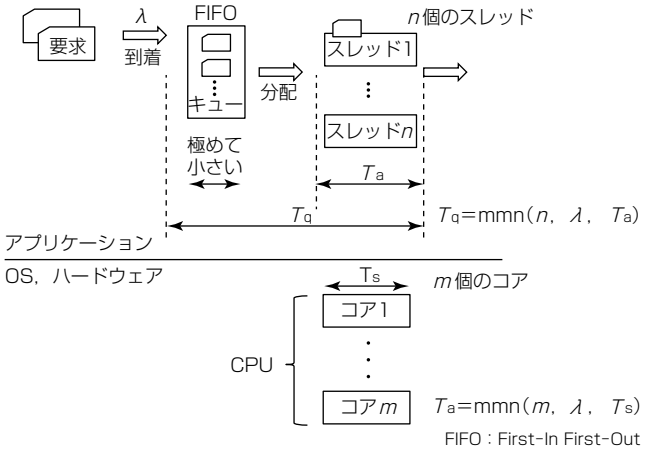


図4．マルチスレッド，マルチコアに対応した性能算出モデル

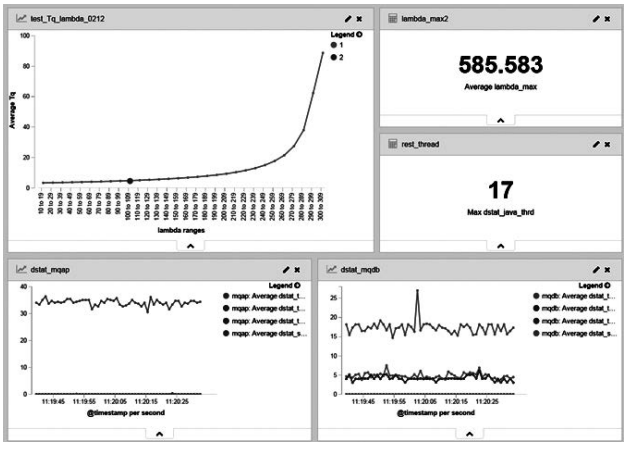


図5．性能評価結果表示の画面例

5.　考慮すべきリソースの特性

計算機リソースサイジングツールは，指定した条件で必要な計算機リソースの台数や，トラフィックが増加したときの応答時間の予測値を提示する。しかし，個々のサーバ計算機やエッジ計算機の能力によってもその必要台数は変わってくる。計算機の構成要素として，性能に影響が大きいCPU，ストレージ，ネットワークを取り上げ，考慮すべき性能特性を次に挙げる。

ツールの出力結果と合わせてこれらの見極めも行う必要がある。

5.1　CPU

計算機のデータ処理は主にCPUで行われる。演算のみの処理の場合，単体での実行時間はほぼCPUを消費した時間であるCPU時間と一致する。

アプリケーションにプロセスやスレッドなど複数のCPUの割当単位がある場合，これらを順次処理することになるので，CPU時間の平均TsとCPU要求の発生密度λ（件/秒）のM/M/1の待ち行列モデルに沿った性能特性となる。

OSが，CPU要求を蓄積するRUNキューを大きくとれば，要求に対する耐力は増すが，それでも平均$1/T_s$を超えることはできない。

マルチコアの場合，CPUの実行を待つRUNキューはそれぞれのコアに存在するが，キュー間の調整も行われるため，複数窓口の待ち行列モデルM/M/nに近似できる（**図6**）。シングルコアのM/M/1は50%の使用率でCPU時間に対して応答時間が平均2倍に劣化するが，マルチコアのM/M/nでは，使用率が上昇しても応答時間の増加の傾向はコア数が多いほど，軽減する。

マルチコアは処理能力をコア数分増加させるだけでなく，高負荷時の性能劣化も軽減する。ただし，仮想化した場合，VM（Virtual Machine）に複数の仮想CPUを割り当て，さらにオーバコミットすると，同時に必要なコア数に関わらず割り付けられた仮想CPU分を物理的なコアを割り付けるため，急速に性能が劣化する。

5.2　ストレージ

ストレージに対するアクセスはシリアライズされるため，性能特性は待ち行列モデルのM/M/1に従う。したがって，その能力の50%を超えるアクセスがあると急速に性能が

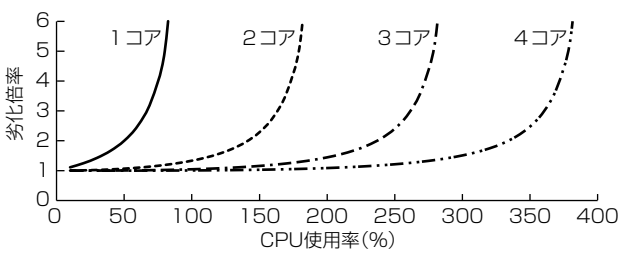


図6．CPU使用率と劣化倍率

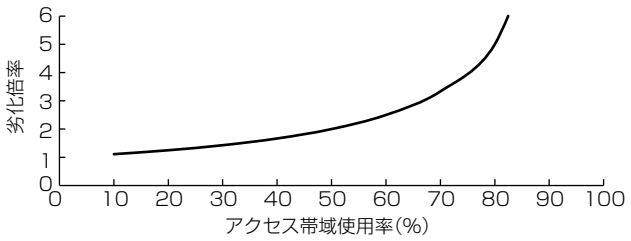


図7．アクセス帯域使用率と劣化倍率

劣化する(**図7**)。

これを回避するためには，より早いアクセス性能を持つストレージを利用する。SSD(Solid State Disk)やIMDB(InMemory DataBase)などの利用は性能向上に効果がある。ネットワークストレージやVMのストレージは，キャッシュによって性能の遅さを軽減しているが，これは散発的なI／Oに対して効果があるだけで，最大負荷時の能力はやはりデバイスの能力に収束する。

5.3 ネットワーク

ネットワークもストレージと同様，シリアライズされるため性能特性は待ち行列モデルのM/M/1に従う。したがって，より広い帯域を持つネットワークを使用する。

計算機上，ネットワークから到着したパケットは，まずはシングルコアで処理されるため，ここがボトルネックとなることもある。短いパケットが多いときや数Gbpsのネットワークとの接続の場合に課題となる。

6. 開発ステップとサイジングツールの活用

計算機リソースサイジング技術は，システムの構想段階から検証，運用開始後まで，計算機リソースの規模と使用状況を予測，計測，評価し，システムの安定した稼働を可能にする。

それぞれのステップでの活用方法について，次に述べる。

6.1 構想・設計

IoTシステムで扱うデータの規模やトラフィックが議論できる程度の段階から，計算機リソースサイジング技術は活用できる。想定している計算機リソースの規模があるならば，計算機リソースサイジングツールを使って，処理可能なトラフィック量を試算することもできる。

また，実績のないパッケージソフトウェアを使用する場合は，何点かのトラフィックの異なる実機計測を行えば，性能特性は把握でき，性能上のリスクは軽減される。

計算機リソースサイジングツールは，検証環境とは異なるCPUコア数の計算機上での実行性能についても予測することができるため，検証は簡易な設備でも行うことができる。

6.2 試験・検証

計算機リソースサイジングツールとエージェント型負荷情報収集ツールと併用することで，試験・検証を効率的に行うことができる。

アプリケーションに手を加えなくても，OSのリソース消費の統計情報だけを使って，性能評価を行うことも可能である。

6.3 稼働・運用

稼働，運用の段階でも，計算機リソースサイジングツールとエージェント型負荷情報収集ツールと併用すると，想定内のトラフィックで稼働しているのか，トラフィックの上昇によって，どの程度応答性能が劣化するのか，ライブで確認することができる。

これによって，IoTシステムの性能劣化が発生する前に，計画的に，計算機リソースの増強などの対策を打つことも可能である。

7. むすび

大規模IoTシステムの計算機リソースのサイジングは，原理的には古典的な大規模システムのサイジングと同様に，単体でのリソース消費量から多重時の応答性能の劣化を予測し，許容範囲に入るまでリソースの量や能力を上げていくことで行うことができる。しかし，IoTシステムを可能とした計算機とネットワークの高度化は，システムの複雑化ももたらし，従来手法では対応しにくい課題も生んでいる。

当社の計算機リソースサイジング技術では，システムの構成要素の還元的な分解，ブラックボックス化された構成要素の性能特性のモデル化，マルチスレッド・マルチコアに対応した応答性能算出，結果が即時に確認できる負荷情報収集を実現することで，大規模IoTシステムにおける計算機リソースサイジングを可能にした。

この技術は，システムの構想段階から検証，運用開始後まで，計算機リソースの使用状況を予測，計測，評価し，システムの安定した稼働を可能とするものである。

参考文献

(1) Molyneaux，I. 著，田中慎司訳：アート・オブ・アプリケーションパフォーマンステスト，オライリージャパン (2009)
(2) 市原利浩，ほか：予備系システムのダウンサイジング手法及び評価，FIT2014(第13回情報科学技術フォーラム)，RO-011 (2014)
(3) Erlang，A.K.：Probability and Telephone Calls，Nyt. Tidsskr. Mat.Ser.B，**20**，33～39 (1909)
(4) Kendall，D.G.：Some Problems in the Theory of Queues，J.Roy.Statist.Soc.，Ser.B，**13**，No.2，151～158 (1951)
(5) Whitt，W.：The Queueing Network Analyzer，THE BELL SYSTEM TECHNICAL JOURNAL，**62**，No.9，2779～2815 (1983)
(6) Lavenberg，S.S.：COMPUTER PERFORMANCE MODELING HANDBOOK，Academic Press (1983)