「マルチメディア，分散，協調とモバイル
(DICOMO2013)シンポジウム」 平成25年7月

クラウドサーバーを用いた携帯端末におけるリソースの最適化

神田正則† 金井謙治† 大木哲史† 甲藤二郎†

近年のスマートフォンやタブレット端末などの携帯端末の著しい普及・高性能化に伴い、大量のデータや計算処理を必要とする、より高度なアプリケーションが求められているが、携帯端末では CPU やメモリ、バッテリーなど様々な制約が多い。そこで、クラウドサーバーへ処理をオフロードすることで時間短縮や省電力化を図る。クラウドサーバーを用いた処理方法では 1) ローカル処理（携帯端末で処理）、2) リモート処理（クラウドサーバーで処理）、3) 並列分散処理（携帯端末とクラウドサーバーで分担して処理）の 3 形態が考えられる。本稿では特に分散並列処理に関して最適となるような配分をモデル化して評価を行った。動画からの顔検出アプリケーションで、今回の環境下では、コーデックに MPEG-4 part2 を用いて最適分配した分散並列処理はリモート処理よりも 20 秒ほどの時間短縮となった。消費電力量はリモート処理の約 3 倍となっているが、ローカル処理と比べると約半分の消費電力で抑えられていることから、ある程度の処理を携帯端末に任せることで処理時間を最小にしつつ、ローカル処理よりも消費電力を抑えることができると言える。

Resource Optimization in the Mobile Terminal using Cloud Server

MASANORI KANDA† KENJI KANAI† TETSUSHI OHKI† JIRO KATTO†

1. はじめに

近年のスマートフォンやタブレット端末などの携帯端末の著しい普及・高性能化に伴い、大量のデータや計算処理を必要とする、より高度なアプリケーション要求が高まっている。しかし、高度なアプリケーションでは大量のデータや計算処理を行うために処理時間や消費電力が増加してしまうことが予想される。携帯端末はバッテリー駆動であることや PC ほどの処理能力を持っていないことからユーザが不満を感じることも少なくない。そこで、携帯端末では負荷のかかりすぎる処理をクラウドサーバーへオフロードすることで、時間短縮や省電力化を図る。クラウドサーバーを用いた処理方法では 1) ローカル処理（携帯端末で処理）、2) リモート処理（クラウドサーバーで処理）、3) 並列分散処理（携帯端末とクラウドサーバーで分担して処理）の 3 形態が考えられる。

本稿では、「高度なアプリケーション要求」と「携帯端末の処理時間および消費電力の増加」というトレードオフの関係に対して、クラウドサーバーを用いることによる処理時間と省電力化の効果を示す。負荷のかかる処理は、すべてをリモートで行うのが望ましいが、リモートに送るためには映像データの圧縮に要する処理量も無視できない。また、携帯端末の高性能化が図られている現在、ある程度の処理を携帯端末で行わせることを考える。そこで、携帯端末とクラウドサーバーでの分散並列処理に関して最適となるような処理配分をモデル化し、実際に Android 端末で負荷のかかるアプリケーションを実装し、処理時間および消費電力の評価を行う。

2. 従来研究

2.1 モデル化

[1]ではクラウドサーバーを用いることで節約できる電力量を以下の式でモデル化を行っている。

$$P_c \times \frac{C}{M} - P_i \times \frac{C}{S} - P_{tr} \times \frac{D}{B} \quad (1)$$

[C: Computation, M: Mobile Power, S: Server Power, D: Data Size, B: Bandwidth, P: Power Consumption (c: compute, i: idle, tr: transfer)]

2.2 Mobile Visual Search

Mobile Visual Search [2]は携帯端末に搭載されているカメラを用いて、物体を撮影し、その写真から物体に関する情報を検索するシステムである。サーバーへの転送は画像か特徴量のみか、物体のマッチング処理はどこで行うのかといったシナリオに関して、消費電力や遅延の評価を行っている。

3. 提案手法 (モデル化)

(1)式は消費電力を重要な指標として考えているが、ユーザによっては時間を重要な指標と考えている場合もある。そこで、データサイズ D [Byte]の処理に、携帯端末では T_{local} [sec], クラウドサーバーでは T_{remote} [sec] の時間がかかるとする。データサイズ D [Byte] のうち、クラウドサーバーで処理するデータサイズを D_{tr} [Byte] とすると、携帯端末で処理を行う時間は

† 早稲田大学基幹理工学研究科情報理工学専攻
Graduate School of Science and Engineering, Waseda University

$$\frac{\mathrm{D}-\mathrm{D_{tr}}}{\mathrm{D}}\times \mathrm{T_{local}} \quad (1)$$

であり、クラウドサーバーで処理を行う時間は、ネットワーク帯域 B [bps] とした場合の転送時間を含めると、

$$\frac{D_{tr}}{D}\times T_{remote}+\frac{2D_{tr}\times 8}{B} \quad (2)$$

と表される。これより、並列分散実行全体に要する時間は

$$\max\left(\frac{D-D_{tr}}{D}\times T_{local},\frac{D_{tr}}{D}\times T_{remote}+\frac{2D_{tr}\times 8}{B}\right)+\alpha \quad (3)$$

で表され（ここでαは分散並列実行処理を行うために必要な処理のオーバーヘッド）、この式が最小値となるように$\mathrm{D_{tr}}$を決定することがよい配分となる。ここで、分散並列実行における携帯端末での処理時間とクラウドサーバーでの処理時間は$\mathrm{D_{tr}}$が増加するにつれ、それぞれ線形的に増加・減少の関係にあるため、(4)式の max 関数の第 1 引数と第 2 引数を等式として解くことで式(4)の最小値が求まる。これより等式を$\mathrm{D_{tr}}$について解くと

$$\mathrm{D_{tr}}=\frac{\mathrm{T_{local}}}{\frac{T_{local}}{D}+\frac{T_{remote}}{D}+\frac{2\times 8}{B}} \quad (4)$$

となる。さらに D で両辺を割ると、

$$\frac{\mathrm{D_{tr}}}{\mathrm{D}}=\frac{\mathrm{T_{local}}}{T_{local}+T_{remote}+\frac{2D\times 8}{B}} \quad (5)$$

となり、これが並列分散処理でクラウドへ転送すべき割合となる。ここで、T_{local}およびT_{remote}は分散並列処理を行う前にあらかじめ実行した結果から値を保持しておく必要がある。

4. 実験

4.1 コンポーネントおよびトポロジー

携帯端末での負荷のかかるアプリケーション例として動画からの顔検出を行わせる Android アプリケーションを実装した。ローカル処理では無圧縮の動画から顔検出処理をかけ、リモート処理では無圧縮動画をエンコードすることでファイルサイズを小さくしてからクラウドサーバーへ転送し、クラウドサーバー内で顔検出処理をかけたものを携帯端末に返す。分散並列処理ではまず、携帯端末用とクラウドサーバー用に動画の分割およびエンコードを行い、分割した動画をそれぞれ顔検出にかけて、最後にそれらの動画を結合するという流れである。実験に用いた動画は 10 [s], フレームレート 30.0 [fps], 解像度 640x480 [pixels] である。実験コンポーネントおよび実験トポロジーはそれぞれ表 1, 図 1 に示すとおりで、デスクトップ PC を仮想的にクラウドサーバーとして用いている。また、End-to-End 帯域は事前に測定した結果から 10.0 [Mbps] 程度であった。

図 1　実験トポロジー

表 1 実験コンポーネント

	Android	Server
CPU	Tegra3 AP33 Max 1.5GHz (Quad Core)	Intel® Core™ i5 M430 @2.27 GHz
Memory	1GB	4GB
OS	Android 4.0.3	Windows 7 32bit

4.2 エンコード

動画のエンコードに関し、Android 端末上で FFmpeg を用いてエンコード処理を行い、それぞれのエンコード方式における処理時間と PSNR (Peak Signal-to-Noise Ratio) を計測した。PSNR は画像の圧縮によりその画像がどの程度劣化したかを評価する指標の 1 つであり、高い方が画質が良いとされている。表 2 から FLV や MPEG-4 part2 は比較的高速にエンコードでき、H.264 はエンコードに多くの処理時間がかかることが確認できる。しかし、図 2 では H.264 はあきらかに他に比べて高い PSNR 値を示しているため、標準的な画質値である 30 [dB]を超えるときのビットレートでは MPEG-4 part2 が 1.5 [Mbit/s] 以上であるのに対し、H.264 では 0.5 [Mbit/s]であるので、同じ画質でもデータサイズを約 3 分の 1 にできるメリットがある。今回の実験では、高速にエンコードでき、比較的高い圧縮率を示す MPEG-4 part2 とエンコードは低速であるが高い圧縮率を示す H.264、の 2 つのコーデックを用いて比較を行う。

表 2 Android 端末上での異なるコーデックにおける
エンコード処理時間(300 フレーム)

CODEC	Time (sec)
MPEG-1	17.1
MPEG-2 / H.262	15.1
H.263+	24.4
FLV (Sorenson H.263)	11.9
MPEG-4 part2	12.3
H264 (x264)	55.7

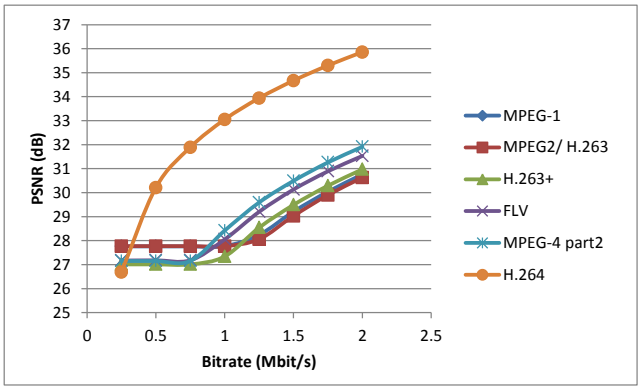

図 2 異なるコーデックにおける PSNR 比較

4.3 処理時間と消費電力量

表 3 に顔検出に必要な処理の所要時間の計測結果を示している。この結果から式(6)で用いるパラメータのT_{local}およびT_{remote}を算出することができる。動画からの顔検出では検出器のロードや初期化処理、パラメータの設定は 1 回だけ実行し、動画から 1 フレーム取り出す作業や顔検出、フレームの符号化などの処理はフレーム数分だけ実行する必要がある。10 秒 300 フレームの動画では、MPEG-4 part2 コーデックを用いた場合に$T_{local} = 224\ [sec]$, $T_{remote} = 88\ [sec]$となり, H.264 コーデックでは$T_{local} = 269$ [sec], $T_{local} = 90\ [sec]$ となる。これより、帯域B = 10.0 [Mbps], データサイズは PSNR 値 30 [dB]を基準として、MPEG-4 part2 の場合はD = 1.99[MB], H.264 の場合はD = 0.60[MB]とし、その他各パラメータを代入すると、処理時間の観点から、MPEG4 part2 ではローカル 29%, リモート 71%, H.264 ではローカル 26%, リモート 74%の割合で分配するもが最もよいという結果となる。

表 3 顔検出に必要な処理の所要時間

Process	Android (ms)	Server (ms)
Read Classifier File	954	788
Initialize	934	2747
Set Parameter	27	146
Decode 1 Frame	6.7	0.99
Gray Scale	5.3	1.01
Down Scale	1.1	0.43
Equalize Histogram	1.1	0.39
Detect Fame	687.7	272.85
Encode 1 Frame	39.09 (MPEG4) 181.37 (H.264)	4.59 (MPEG4) 10.26 (H.264)

Android 端末で顔検出処理の時間を計測した結果を図 4 に示す。この図より式(6)から求めた値に近いところで、分散並列処理の最小値になっていることが確認できる。特に MPEG-4 part2 では動画の分割・結合などのオーバーヘッドを含めてもリモート処理よりも 20 秒ほど処理時間が短縮できていることが確認できる。一方、H.264 ではエンコードにかかるオーバーヘッドの時間がかなり大きくなってしまっていることから、リモート処理よりも時間が短縮できないという結果になった。今回の実験では、帯域がある程度の大きさがあるためにデータ転送にかかる時間がごく微量となっているが、帯域が小さい場合や動画を高画質な状態で転送する場合などには、圧縮率の高い H.264 コーデックの有効性が表れると考えられる。なお、図 3 に示している理論値は携帯端末の処理時間とデータ転送を含めたクラウドサーバーでの処理時間の MAX 関数をとったものであり、動画の結合・分割などのオーバーヘッドを考慮していないため、実際の実験結果よりも小さい値となっている。

また、電力モデルを[2]から推定して CPU 使用時の電力を$P_c = 0.4$ [W], アイドル時の電力 $P_i = 0.05$ [W],データ転送時の電力 $P_{tr} = 0.75$ [W] として算出した消費電力量を図 4 に示す。MPEG-4 part2, H.264 のともに、最適分配に近いローカル 30%, リモート 70%の並列分散処理では、リモート処理に比べて約 3 倍の消費電力量となってしまっていることが確認できる。しかし、ローカル処理に比べては約半分の消費電力量ですむという結果であった。今回の環境下では、ローカル処理する割合が増えるにつれ消費電力量が増加していく傾向であったが、帯域が極端に狭い場合には、ローカル処理する割合が増えるにつれ消費電力量が減少していく傾向になるため、分散並列処理は常にローカル処理とリモート処理の中間程度の消費電力量となると考えられる。

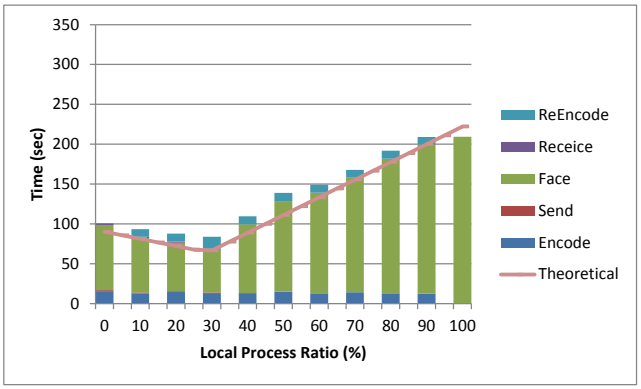

図 3.(a) 処理時間 (MPEG-4 part2)

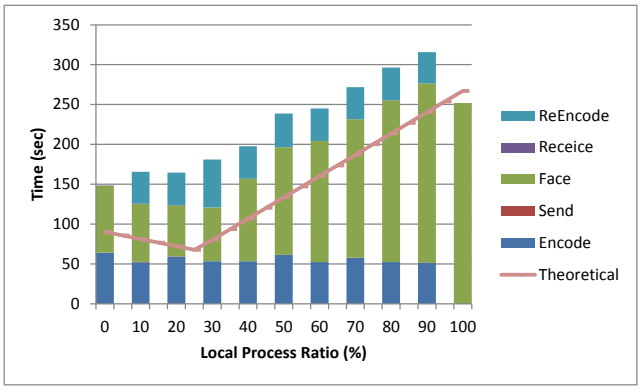

図 3.(b) 処理時間 (H.264)

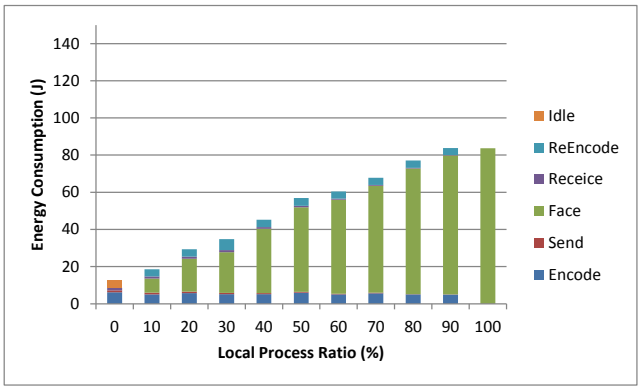

図 4.(a) 消費電力量 (MPEG-4 part2)

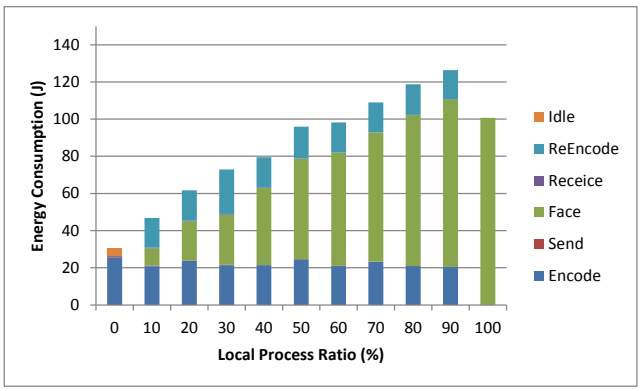

図 4.(b) 消費電力量 (H.264)

4.4 CPU 使用率およびメモリ使用量

Android 端末での各処理方法に対する CPU 使用率およびメモリ消費量に関する調査を行った。その結果を図 5, 図 6 に示す。図 5 から、ローカル処理は常に高い CPU 使用率を示すのに対し、リモート処理ではほぼ 0 の値を示していることが確認できる。この図の分散並列処理は携帯端末での処理がクラウドサーバーでの処理よりも早く終わるような分配をしたときの結果であり、携帯端末での処理が終わるとアイドル状態になるので、急激に CPU 使用率が減少していることが確認できる。なお、CPU 使用率が 100%に達しないのはシステムの動作やバックグラウンド動作で CPU を消費しているからだと考えられる。また、メモリ使用量に関しては、ローカル処理および並列分散処理に対してリモート処理ではメモリ使用量が 4 分の 1 程度となっていることが確認できる。これは顔検出に必要となるオブジェクトをメモリに読み出す必要がなくなるためである。このことから、CPU 負荷やメモリ使用量に制約がある場合は、リモート処理が最も有効となることが考えられる。

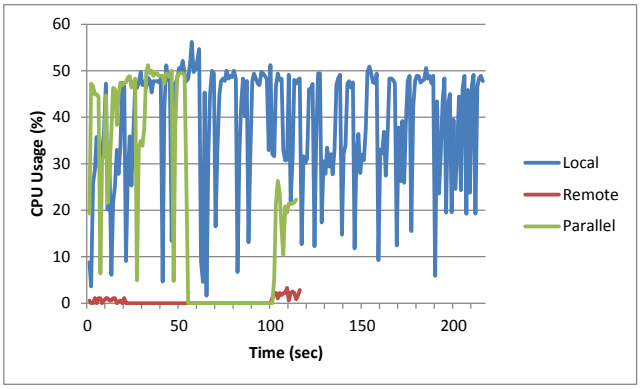

図 5 Android 端末における CPU 使用率の比較

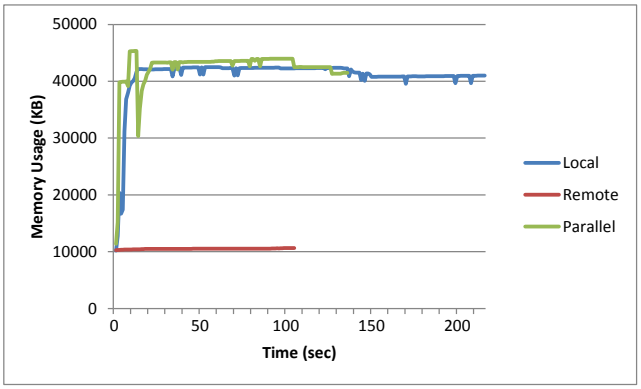

図 6 Android 端末におけるメモリ使用量の比較

5. まとめと今後の展望

クラウドサーバーを用いることで、携帯端末における負荷のかかる処理を転送し、短時間・省電力で実行することができる。あらかじめ携帯端末およびクラウドサーバーでの処理時間の見積もりがとれれば、携帯端末の高性能化により、ある程度の処理を携帯端末で行わせるように最適に分散することで、並列分散処理がリモート処理よりもさらに短い時間で処理を終えることも可能である。しかし、CPU 負荷やメモリ使用に制約がある場合にはかならずしもこの通りになるとは言えない。今回は Android 端末およびクラウドサーバーが軽負荷時の状態で実験を行ったが、クラウドサーバーがほかの処理を行っている状況のほうが一般的であるため、今後、高負荷な状態における評価や、画像検索[2]やオンラインゲーム、三次元構造復元など、各種のマルチメディア応用に関する検討を進めていく。

参考文献

1) K.Kumar and Y.-H.Lu: “Cloud computing for mobile users: Can Offloading Computation Save Energy?” IEEE Computer, Vol.43, pp.51-56, Apr.2010.
2) B.Girod et al.: “Mobile Visual Search,” IEEE Signal Processing Magazine, Vol.28, pp.61-76, Jul.2011
3) L.Zhang and B.Wiwana: “Accurate Online Power Estimation and Automatic Battery Behavior Base Power Model Generation for Smartphones” IEEE/ACM/IFIP CODES+ISSS 2010, pp105-114, Oct.2010.
4) “javaCV” http://code.google.com/p/javacv/
5) “FFmpeg” http://ffmpeg.org/
6) “OpenCV” http://opencv.org/