

修士論文

データベース管理システムにおける再起動時の  
問合せ実行待ち時間の軽減に関する研究

平成30年07月12日提出

指導教員 喜連川 優 教授

情報理工学系研究科 電子情報学専攻

48-146427 谷川 祐一

## 概要

データベース管理システムは、障害等による再起動時に、最新のチェックポイントに遡ってログを適用し、一貫性を確保してから問合せ処理を開始する。早期ログ適用はログ適用と問合せ処理を並行して実行する技法であり、これにより再起動時の問合せ処理の待ち時間短縮が期待される。本研究では当該効果を試作データベースエンジンを用いた実験により明らかにする。

# 目次

1	はじめに	1
2	データベース管理システムにおける回復可能性と再起動過程	3
2.1	データベース管理システムの回復可能性	3
2.2	データベース管理システムの再起動過程	3
3	早期ログ適用技法を用いた再起動過程	6
4	実験用データベースエンジンの実装	8
4.1	試作データベースエンジンの機能と動作	8
4.1.1	単純ログ適用機能	8
4.1.2	単純問合せ処理機能	9
4.1.3	早期ログ適用を用いた問合せ処理機能	9
4.2	試作データベースエンジンの実装の詳細	10
4.2.1	共有バッファ	10
4.2.2	ログリーダー・ログパッチャ	10
4.2.3	問合せ実行器	11
4.3	データベースとログの物理構造	11
4.3.1	データベースボリューム	11
4.3.2	ログボリューム	11
5	早期ログ適用技法による問合せ実行待ち時間の軽減効果の評価	12
5.1	実験環境	12
5.2	想定データセットと負荷	12
5.2.1	テストログ	13
5.2.2	テスト問合せ	13
5.3	単純ログ適用の基本性能	13
5.4	単純問合せ実行の基本性能	21
5.5	データベース管理システムの再起動時の問合せ実行待ち時間	56
6	まとめと今後の課題	65

# 目次

1	ライトアヘッドロギングとチェックポイントによるトランザクションの永続化	4
2	データベース管理システムの再起動過程	5
3	早期ログ適用技法を用いたデータベース管理システムの再起動過程	6
4	試作データベースエンジンのコンポーネント概略図	8
5	単純ログ適用の実行時間	14
6	単純ログ適用の発行 I/O 数	15
7	単純ログ適用の平均 I/O スループット	15
8	単純ログ適用時の毎秒 CPU 利用率・I/O スループット (Rand・逐次ログ適用)	16
9	単純ログ適用時の毎秒 CPU 利用率・I/O スループット (Rand・整列ログ適用)	16
10	単純ログ適用時の毎秒 CPU 利用率・I/O スループット (80/20・逐次ログ適用)	17
11	単純ログ適用時の毎秒 CPU 利用率・I/O スループット (80/20・整列ログ適用)	17
12	単純ログ適用時の毎秒 CPU 利用率・I/O スループット (90/10・逐次ログ適用)	18
13	単純ログ適用時の毎秒 CPU 利用率・I/O スループット (90/10・整列ログ適用)	18
14	単純ログ適用時の毎秒 CPU 利用率・I/O スループット (99/1・逐次ログ適用)	19
15	単純ログ適用時の毎秒 CPU 利用率・I/O スループット (99/1・整列ログ適用)	19
16	単純ログ適用時の毎秒 CPU 利用率・I/O スループット (99.9/0.1・逐次ログ適用)	20
17	単純ログ適用時の毎秒 CPU 利用率・I/O スループット (99.9/0.1・整列ログ適用)	20
18	単純問合せ実行の実行時間	21
19	単純問合せ実行の発行 I/O 量	22

20	単純問合せ実行の平均 I/O スループット	23
21	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Seq・ $\sigma = 0.1$ )	23
22	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Seq・ $\sigma = 0.2$ )	24
23	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Seq・ $\sigma = 0.3$ )	24
24	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Seq・ $\sigma = 0.4$ )	25
25	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Seq・ $\sigma = 0.5$ )	25
26	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Seq・ $\sigma = 0.6$ )	26
27	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Seq・ $\sigma = 0.7$ )	26
28	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Seq・ $\sigma = 0.8$ )	27
29	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Seq・ $\sigma = 0.9$ )	27
30	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Seq・ $\sigma = 1$ )	28
31	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand・ $\sigma = 1e-6$ )	28
32	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand・ $\sigma = 2e-6$ )	29
33	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand・ $\sigma = 3e-6$ )	29
34	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand・ $\sigma = 4e-6$ )	30
35	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand・ $\sigma = 5e-6$ )	30
36	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand・ $\sigma = 6e-6$ )	31

37	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand・ $\sigma = 7e-6$ )	31
38	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand・ $\sigma = 8e-6$ )	32
39	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand・ $\sigma = 9e-6$ )	32
40	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand・ $\sigma = 1e-5$ )	33
41	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand・ $\sigma = 2e-5$ )	33
42	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand・ $\sigma = 3e-5$ )	34
43	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand・ $\sigma = 4e-5$ )	34
44	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand・ $\sigma = 5e-5$ )	35
45	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand・ $\sigma = 6e-5$ )	35
46	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand・ $\sigma = 7e-5$ )	36
47	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand・ $\sigma = 8e-5$ )	36
48	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand・ $\sigma = 9e-5$ )	37
49	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand・ $\sigma = 1e-4$ )	37
50	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand・ $\sigma = 2e-4$ )	38
51	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand・ $\sigma = 3e-4$ )	38
52	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand・ $\sigma = 4e-4$ )	39

53	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand・ $\sigma = 5e-4$ ) . . . . .	39
54	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand・ $\sigma = 6e-4$ ) . . . . .	40
55	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand・ $\sigma = 7e-4$ ) . . . . .	40
56	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand・ $\sigma = 8e-4$ ) . . . . .	41
57	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand・ $\sigma = 9e-4$ ) . . . . .	41
58	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand・ $\sigma = 1e-3$ ) . . . . .	42
59	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand・ $\sigma = 2e-3$ ) . . . . .	42
60	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand・ $\sigma = 3e-3$ ) . . . . .	43
61	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand・ $\sigma = 4e-3$ ) . . . . .	43
62	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand・ $\sigma = 5e-3$ ) . . . . .	44
63	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand・ $\sigma = 6e-3$ ) . . . . .	44
64	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand・ $\sigma = 7e-3$ ) . . . . .	45
65	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand・ $\sigma = 8e-3$ ) . . . . .	45
66	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand・ $\sigma = 9e-3$ ) . . . . .	46
67	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand・ $\sigma = 0.01$ ) . . . . .	46
68	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand・ $\sigma = 0.02$ ) . . . . .	47

69	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand・ $\sigma = 0.03$ )	47
70	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand・ $\sigma = 0.04$ )	48
71	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand・ $\sigma = 0.05$ )	48
72	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand・ $\sigma = 0.06$ )	49
73	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand・ $\sigma = 0.07$ )	49
74	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand・ $\sigma = 0.08$ )	50
75	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand・ $\sigma = 0.09$ )	50
76	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand・ $\sigma = 0.1$ )	51
77	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand・ $\sigma = 0.2$ )	51
78	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand・ $\sigma = 0.3$ )	52
79	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand・ $\sigma = 0.4$ )	52
80	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand・ $\sigma = 0.5$ )	53
81	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand・ $\sigma = 0.6$ )	53
82	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand・ $\sigma = 0.7$ )	54
83	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand・ $\sigma = 0.8$ )	54
84	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand・ $\sigma = 0.9$ )	55



85	単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand・ $\sigma = 1$ )	55
86	再起動時の問合せ実行待ち時間 (ログ: Rand) (問合せ: Rand・ $\sigma = 1e-5$ )	57
87	再起動時の問合せ実行待ち時間 (ログ: Rand) (問合せ: Rand・ $\sigma = 1e-6$ )	58
88	従来の再起動過程の毎秒 CPU 利用率・I/O スループット (ログ: Rand・逐次ログ適用) (問合せ: Rand・ $\sigma = 1e-5$ )	58
89	従来の再起動過程の毎秒 CPU 利用率・I/O スループット (ログ: Rand・整列ログ適用) (問合せ: Rand・ $\sigma = 1e-5$ )	59
90	早期ログ適用を用いた再起動過程の毎秒 CPU 利用率・I/O スループット (ログ: Rand・逐次ログ適用) (問合せ: Rand・ $\sigma = 1e-5$ , 同時受付)	59
91	早期ログ適用技法を用いた再起動過程の毎秒 CPU 利用率・I/O スループット (ログ: Rand・整列ログ適用) (問合せ: Rand・ $\sigma = 1e-5$ , 同時受付)	60
92	早期ログ適用技法を用いた再起動過程の毎秒 CPU 利用率・I/O スループット (ログ: Rand・逐次ログ適用) (問合せ: Rand・ $\sigma = 1e-5$ , 50 秒後受付)	60
93	早期ログ適用技法を用いた再起動過程の毎秒 CPU 利用率・I/O スループット (ログ: Rand・整列ログ適用) (問合せ: Rand・ $\sigma = 1e-5$ , 50 秒後受付)	61
94	従来の再起動過程の毎秒 CPU 利用率・I/O スループット (ログ: Rand・逐次ログ適用) (問合せ: Rand・ $\sigma = 1e-6$ )	61
95	従来の再起動過程の毎秒 CPU 利用率・I/O スループット (ログ: Rand・整列ログ適用) (問合せ: Rand・ $\sigma = 1e-6$ )	62
96	早期ログ適用技法を用いた再起動過程の毎秒 CPU 利用率・I/O スループット (ログ: Rand・逐次ログ適用) (問合せ: Rand・ $\sigma = 1e-6$ , 同時受付)	62
97	早期ログ適用技法を用いた再起動過程の毎秒 CPU 利用率・I/O スループット (ログ: Rand・整列ログ適用) (問合せ: Rand・ $\sigma = 1e-6$ , 同時受付)	63

98	早期ログ適用技法を用いた再起動過程の毎秒 CPU 利用率・I/O スループット (ログ: Rand・逐次ログ適用) (問合せ: $\text{Rand} \cdot \sigma = 1e-6$ , 50 秒後受付) . . . . .	63
99	早期ログ適用技法を用いた再起動過程の毎秒 CPU 利用率・I/O スループット (ログ: Rand・整列ログ適用) (問合せ: $\text{Rand} \cdot \sigma = 1e-6$ , 50 秒後受付) . . . . .	64

# 1 はじめに

データベース管理システムは、大規模データをデータベースとして編成して堅牢に管理し、また、これに対する問合せ等の処理を迅速に実行するためのソフトウェアである。多くのデータベース管理システムでは、障害等からの回復可能性を実現するために、ライトアヘッドロギングならびにチェックポイントなる制御手法を採用している [1][2][3]。ライトアヘッドロギングは、データベースへの更新をログとして書き出すものである。すなわち、データベース管理システムは、更新がログに永続的に記録されたことをもって、更新トランザクションが永続化されたとみなし、これをコミットする。この際、データベースの更新が実際にデータベースを格納するボリュームに永続的に記録されたかどうか、もしくは、データベースバッファ中にダーティーな状態で滞留しているかどうかは問わない。対してチェックポイントは、データベースバッファ中に滞留している更新のうち、コミットされたトランザクションによるものを、データベースを格納するボリュームに永続的に記録するものである。すなわち、チェックポイントが発動された時点において既にコミットされているトランザクションによる更新は、チェックポイントが終了した時点で全てボリュームに永続的に記録されていることになる。

データベース管理システムは、障害等により再起動されると、データベースを格納するボリュームならびにログのディレクトリ情報等を探索し、最新のチェックポイントを同定し、当該チェックポイント以降に記録されたログをデータベースに適用し、データベースを回復し、一貫性のある状態となってから問合せ処理を受け付け、これの処理を実行する。よって、データベース管理システムに障害等が生じ再起動される場合、データベースを回復するまでの間、データベース管理システムは新たな問合せを受け付けて処理することができない。これはデータベース管理システムの可用性を低下させる1つの要因となっている。

早期ログ適用は、データベース管理システムが障害等により再起動された際に、ログの適用によるデータベースの回復を待つことなく、先行的に問合せ処理を実行可能とする技法である [4]。ログ適用中のデータベースは、既にコミットされたトランザクションによる更新が全て適用されていることが保証されない。よって、問合せがデータベースからページを取得する際には、当該ページが既に回復済みであって既にコミットされたトランザクションによる全ての更新が適用されているかどうかを判定し、そうであれば当該ページをそのまま利用し、そうでない場

合には当該ページに未適用のログを取得することによりコミットされたトランザクションによる全ての更新が適用されている状態に回復させてから当該ページを利用する。当該技法によれば，データベース管理システムに障害等が生じ再起動される場合，データベースを回復するまで待つことなく，新たな問合せを受け付けて処理することができるようになり，これによってデータベース管理システムの可用性を向上することが期待される。

本論文は，早期ログ適用によってログ適用中に発行された問合せ処理の実行待ち時間が改善する効果を，実験により明らかにする。著者は，早期ログ適用機能を備えたデータベースエンジンを試作し，複数の更新偏りパターンの更新トランザクションによって生成された一連のログを対象として，当該ログの適用中に複数のデータアクセス偏りパターンを有する問合せを発行し，それぞれの組合せで，ログ適用ならびに問合せ処理に要する実行時間や実行中に各種資源利用量を計測し，問合せ処理の実行待ち時間の変化を評価した。当該評価を本論文ではまとめ，評価の結果，早期ログ適用技法を用いることにより，ログ適用中に発行された問合せ処理の実行待ち時間が改善することを明らかにする。

## 2 データベース管理システムにおける回復可能性と再起動過程

### 2.1 データベース管理システムの回復可能性

データベース管理システムは大規模なデータを堅牢に管理して、それに対する問合せ処理を迅速に実行するためのソフトウェアである。多くのデータベース管理システムは、ライトアヘッドロギングとチェックポイントによって障害からの回復可能性を実現している。

ライトアヘッドロギングは、データベースへの更新を実際の更新に先駆けてログとして書き出すもので、ログが永続的に記録されたことをもって、トランザクションによる更新が永続化されたとみなす。チェックポイントは、永続化されたトランザクションの更新を遅れてデータベースに書き込むもので、チェックポイントの終了時点においてトランザクションによる更新がデータベースに永続的に記録されていることになる。

図1はデータベースへの更新がディスクに永続化される様子を表した概念図である。横軸を時間として、トランザクションによる更新が、データベースボリュームとログボリュームに記録される時間の様子を矢印として表している。ライトアヘッドロギングによって、トランザクションによる更新はログボリュームには即座に記録されるが、データベースボリュームに実際に記録されるのが保証されるのは次のチェックポイントの終了時点になる。

### 2.2 データベース管理システムの再起動過程

データベース管理システムの障害からの再起動時には、データベースボリュームから最新のチェックポイントを探し出し、それに対してチェックポイント以後のログを適用することで、データベースを一貫性のある状態に回復する。このとき一貫性のある状態に回復してから初めて問合せ処理の実行を開始する。よってデータベース管理システムの再起動時には、問合せ処理の実行を開始するまでにログの適用待ち時間が存在し、これが可用性低下の一要因になっている。

図2はデータベース管理システムの障害からの再起動過程の様子を表した概念

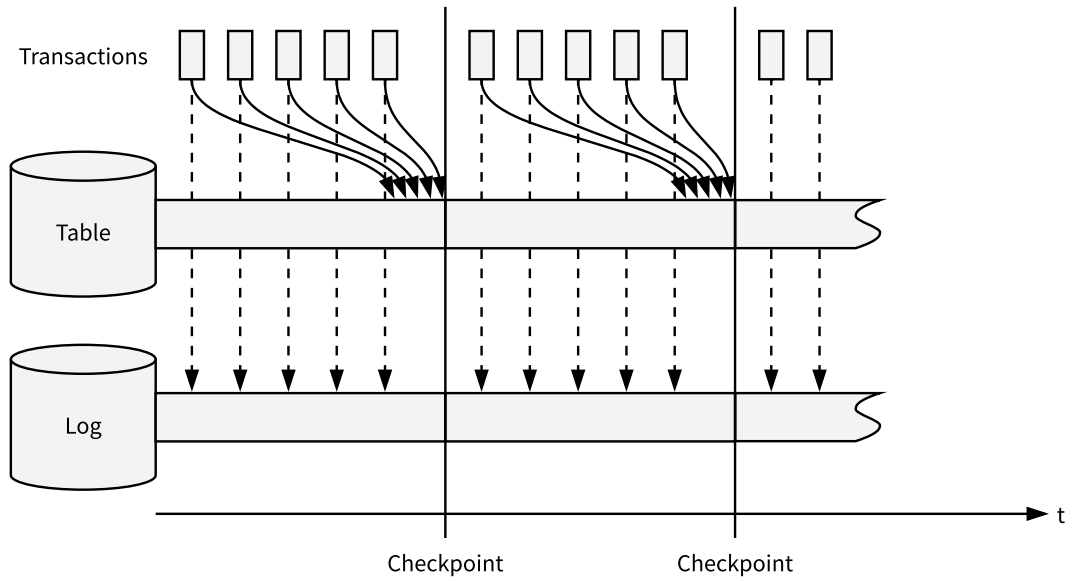


図 1: ライトアヘッドロギングとチェックポイントによるトランザクションの永続化

図である。大きなバツ印の時点で障害が起こったとする。障害時点で最新のチェックポイント以後の更新はログボリュームには記録されているが、データベースボリュームに記録されていることは保証されていない。再起動時には、データベースボリュームの最新チェックポイントに対して、それ以後のログを適用することで障害時点の状態に回復する。このとき、回復のためのログ適用中は問合せ処理を実行することができず、ログ適用の完了を待つ必要がある。

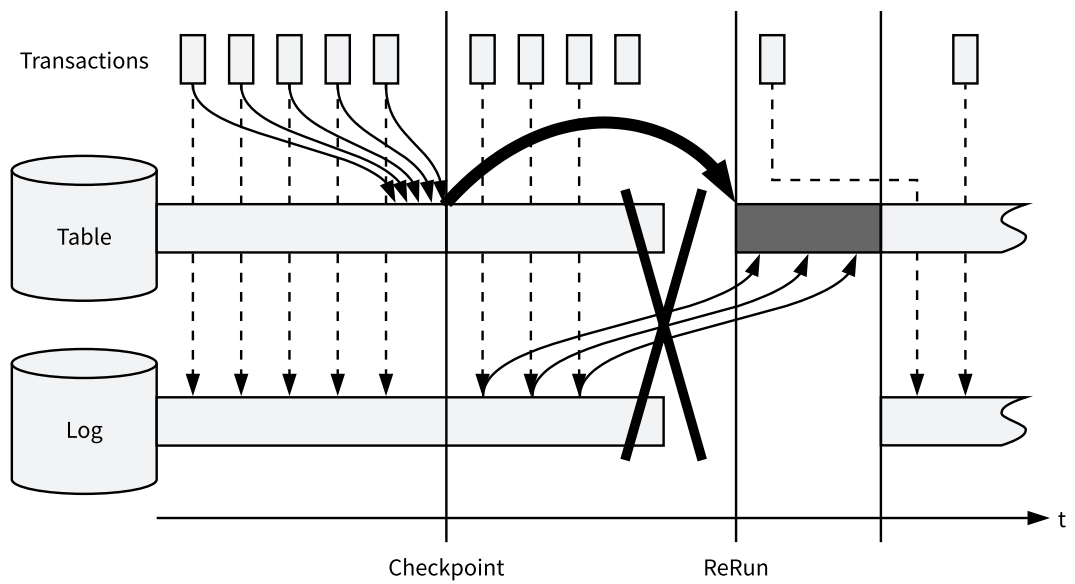


図 2: データベース管理システムの再起動過程

### 3 早期ログ適用技法を用いた再起動過程

早期ログ適用技法とは，データベース管理システムの再起動時にログ適用によるデータベースの回復を待つことなく，問合せ処理を実行可能にする仕組みである．問合せ処理がデータベースからレコードを読み出すときに，必要に応じて未適用のログを取得してそれを適用したレコードを利用して問合せを処理する．これによってログ適用中においても，ログ適用後の一貫性のあるデータベースに対する問合せ応答を実現できる．

図3は早期ログ適用技法を用いたデータベース管理システムの障害からの再起動過程の様子を表した概念図である．従来の再起動過程である図2と比較すると，従来では存在した再起動時のログ適用待ち時間が存在せず，問合せ処理を即座に実行できていることが確認できる．

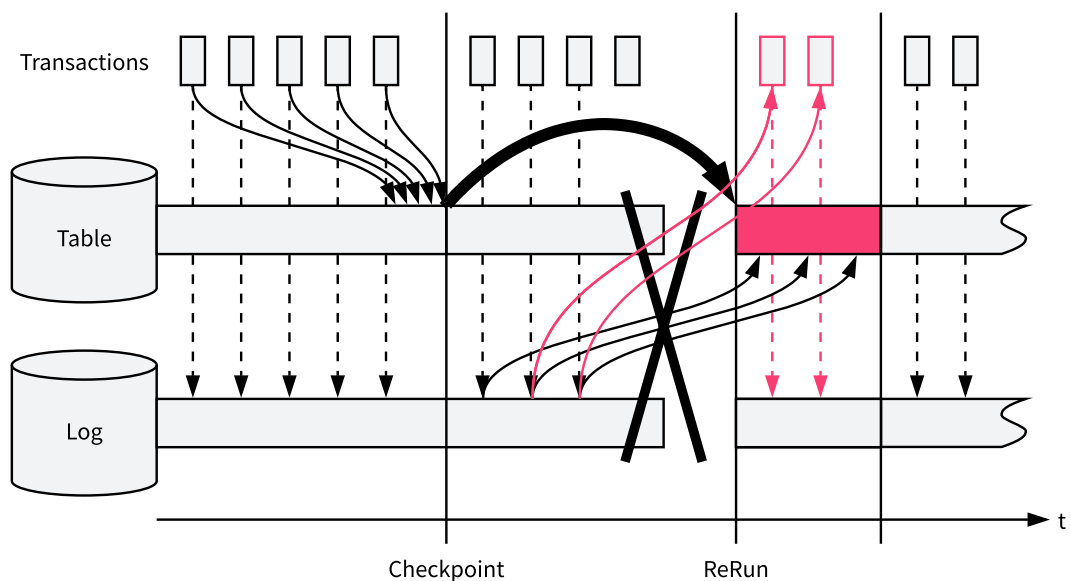


図 3: 早期ログ適用技法を用いたデータベース管理システムの再起動過程

具体的には，早期ログ適用技法は，以下の手順を行う．まず，ログ適用中のデータベースは，既にコミットされたトランザクションによる更新が全て適用されていることが保証されない．よって，問合せがデータベースからページを取得する



際には、当該ページが既に回復済みであって既にコミットされたトランザクションによる全ての更新が適用されているかどうかを判定し、そうであれば当該ページをそのまま利用し、そうでない場合には当該ページに未適用のログを取得することによりコミットされたトランザクションによる全ての更新が適用されている状態に回復させてから当該ページを利用する。このような手順によって、データベース管理システムに障害等が生じ再起動される場合、データベースを回復するまで待つことなく、新たな問合せを受け付けて処理することができるようになり、これによって、データベース管理システムの可用性を向上することが期待される。

この際、早期ログ適用技法の適用によって、どれほど問合せ実行待ち時間が軽減するのかを実験によって定量的に把握することが、本論文の目的である。具体的には、第一に、データベースに対する異なる偏りアクセスから生じたログ系列について、ログ適用にかかる時間を計測する。この際、ログが生成された順序でそのまま適用することに加え、ログをアクセス先アドレスの順序に並び替えて適用することによるログ適用効果を検証する。第二に、複数の問合せアクセスパターンに対して、問合せ処理にかかる時間を計測する。これらをもとに、第三に、ログ適用を実行しながら問合せ処理を行った場合に、早期ログ適用を行った場合と行わない場合を比較し、早期ログ適用による問合せ実行待ち時間の軽減効果を把握する。この際、早期ログ適用を行わず整列ログ適用によって単にログ適用にかかる時間を短縮することによる副次的な問合せ実行待ち時間の軽減効果と、早期ログ適用による問合せ実行待ち時間の軽減効果を比較することにより、早期ログ適用による優位性を明らかにする。続く4章以降で、これらの実験の詳細を述べる。

## 4 実験用データベースエンジンの実装

早期ログ適用技法による効果を実験によって検証するために、早期ログ適用技法を備えたデータベースエンジンを試作した。本章では試作したデータベースエンジンの機能とその実装について述べる。

### 4.1 試作データベースエンジンの機能と動作

試作データベースエンジンは単純ログ適用機能、単純問合せ処理機能、早期ログ適用を用いた問合せ処理機能の3つの機能を備える。また、試作データベースエンジンのコンポーネントの概略図を図4に示す。

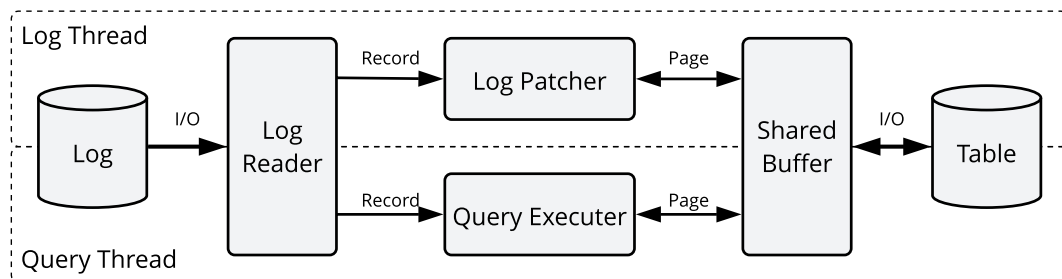


図 4: 試作データベースエンジンのコンポーネント概略図

#### 4.1.1 単純ログ適用機能

1つめの機能は、ログ適用を単体を実行する単純ログ適用機能である。その動作としては、ログボリュームからログを読み出して、データベースに適用するというものである。

疑似アルゴリズムを Listing 1 に示す。ログボリュームからログレコードを読み出し、対象行を含むページのロックを共有バッファから取得し、対象行を更新する。

Listing 1: 単純ログ適用の疑似アルゴリズム

```
1 Foreach record in LogVolume:
2     page := SharedBuffer.lock(record.row_id)
3     page.update(record)
```

#### 4.1.2 単純問合せ処理機能

2つめは、問合せ処理を単体で実行する単純問合せ処理機能である。その動作としては、データベースからレコードを読み出して、問合せを処理するというものである。

疑似アルゴリズムを Listing 2 に示す。選択率  $\sigma$  に達するまで、アクセスパターンに基づいて次に参照する行 ID を生成し、対象行を含むページのロックを共有バッファから取得し、対象行を参照する。

Listing 2: 単純問合せ処理の疑似アルゴリズム

```
1 Repeat until  $\sigma$  :  
2     row_id := AccessPattern.next()  
3     page := SharedBuffer.lock(row_id)  
4     row := page.row[row_id]
```

#### 4.1.3 早期ログ適用を用いた問合せ処理機能

3つめは、早期ログ適用技法によってログ適用と並行して実行する問合せ処理機能である。この時の問合せ処理の動作は、データベースからレコードを読み出し、必要に応じてログボリュームから未適用のログを読み出して適用して、問合せを処理するというものである。

問合せ処理の疑似アルゴリズムを Listing 3 に示す。選択率  $\sigma$  に達するまで、アクセスパターンに基づいて次に参照する行 ID を生成する。このとき、対象行を対象とするログレコードがログボリューム中に存在する場合は、ログボリュームからログレコードを読み出し、対象行を参照したとする。存在しない場合は単純問合せ処理と同じく、対象行を含むページのロックを共有バッファから取得し、対象行を参照する。

Listing 3: 早期ログ適用を用いた問合せ処理の疑似アルゴリズム

```
1 Repeat until  $\sigma$  :  
2     row_id := AccessPattern.next()  
3     If row_id in LogVolume:  
4         record := LogVolume.get(row_id)  
5         row := record.row
```

```
6 Else :
7     page := SharedBuffer.lock(row_id)
8     row := page.row[row_id]
```

データベース管理システムの再起動過程を模擬するときには、従来技法であるログ適用を待って問合せ処理を開始するする再起動過程として、単純ログ適用機能の後に単純問合せ処理機能を実行する。早期ログ適用技法を用いた再起動過程を模擬するときには、単純ログ適用機能と早期ログ適用技法を用いた問合せ処理を並行して別スレッドで実行する。

## 4.2 試作データベースエンジンの実装の詳細

### 4.2.1 共有バッファ

ログ適用スレッドと問合せ処理スレッドにおいて共有されるページバッファである。バッファ管理アルゴリズムとして Least Recent Used を採用した。ページ単位の排他ロックを用いて、ログ適用と問合せ実行の並列実行時の排他制御を行う。バッファプールのサイズ、すなわちバッファ中に保持するページ数は可変としたが、バッファプールのサイズを変えた時の性能の変化を見るのは今回の実験の目的からは外れているので、実験においてはバッファプールのサイズは一律 128 ページとした。

### 4.2.2 ログリーダー・ログパッチャ

ログリーダーはログボリュームから一度に全てログを読み込み、ログパッチャはログリーダーが読み込んだレコードを順番にデータベースボリュームへと適用する。ログを適用するときの順番として、逐次ログ適用と整列ログ適用の 2 通りのログ適用手法を用意した。

**逐次ログ適用** ログが生成された順番そのまま、すなわちログリーダーがログを読み込んだ順番そのままに、ログを適用する。単純で広く用いられる手法である。

**整列ログ適用** ログを、その適用先レコードの、データベースボリューム中のテーブルにおけるアドレス順に適用する手法である。これはログ適用における I/O の連続性を高める改善手法で、商法システムなどで利用されている。ログリーダーが、ログボリュームからログを読み込んだ後に、ログを主記憶上でテーブルのアドレス順に並べ替え、その後ログパッチャがログ適用を行うこととした。

早期ログ適用技法を用いる場合、問合せ処理の対象となるレコードに対するログがログボリュームに存在するかどうかを判定する必要がある。そのため、ログリーダーはログの読み込み後（整列ログ適用の場合は、ログの並べ替え後）、ログの適用先レコード ID の索引を作成する。

### 4.2.3 問合せ実行器

問合せ実行器は、任意の SQL による問合せ処理を実行するというものではなく、テーブルに対するアクセスパターンと、テーブル中の全レコードに対する読み出すレコードの割合である選択率を指定して、問合せ処理を実行するものとした。

## 4.3 データベースとログの物理構造

### 4.3.1 データベースボリューム

データベースボリュームは単一のテーブルで構成される。ページは 8KB、レコードは固定長で 12B とした。索引は省き、一意なレコード ID（一意なページ ID とページ内オフセットの組）によってレコードおよびそのレコードを含むページにアクセスする。

### 4.3.2 ログボリューム

1つのログレコードは、データベースボリュームの 1レコードに対する変更を表す。ログレコードは固定長であり、更新ログについては、UNDO と REDO を行うために、更新前のデータと更新後のデータを含む。

## 5 早期ログ適用技法による問合せ実行待ち時間の軽減効果の評価

試作したデータベースエンジンを用いて、単純ログ適用、単純問合せ処理、そして早期ログ適用技法による問合せ処理について実験を行った。本章では実験環境、実験設定、実験結果について述べる。

### 5.1 実験環境

実験を行った計算機のハードウェア・ソフトウェア環境は表1の通りである。データベースエンジンのプロセスを実行するCPUについては、numactlを利用して単一のスレッドにのみ割り当てるようにした。また発行するI/Oについては、ダイレクトI/Oを利用してシステムのページキャッシュをバイパスするようにした。

表 1: 実験環境

CPU	Intel Xeon E5-2690 v2 @ 3.00GHz
# of sockets, cores, threads	2, 20, 40
Memory	64GB
OS	CentOS 6.9 (Final)
Kernel	2.6.32-696.20.1.el6
HDD	900GB (10000RPM SAS)
Filesystem	XFS
テーブル用ストレージ	HDD x8 RAID 6
ログ用ストレージ	HDD x2 RAID 1

### 5.2 想定データセットと負荷

初期データベースとして単一テーブルからなる100GB（13百万ページ、6.7十億行）のデータベースを作成した。レコード長は固定長で12Bである。問合せ処理とログ適用は全てこのテーブルを対象とする。

### 5.2.1 テストログ

実験に用いるテストログは、データベースレコードの更新のみを扱うこととし、挿入や削除は考慮しないこととした。テストログのサイズは4.6MiB（レコード数としては100千レコード）を用意した。

ログを生成するパターンとしては、テーブルへのランダムアクセスを想定して、偏りのないランダムアクセスと、いくつかの偏りの異なるランダムアクセスを用意した。偏りのないランダムアクセスをRandと表すこととし、これはテーブルの全レコードを無作為に選んで更新した場合とみなすことができる。他方、偏りのあるランダムアクセスをX/Yと表すこととし、これはログの内X%のアクセスがテーブルの全レコードの内Y%を無作為に選んで更新した場合とみなすことができる。このX/Yのパターンとしては、偏りの異なる80/20, 90/10, 99/1, 99.9/0.1の4通りを用意した。

またそれぞれのログを適用するときには、逐次ログ適用と整列ログ適用の2通りのログ適用手法を用いた。

### 5.2.2 テスト問合せ

問合せはデータベースレコードの選択のみとし、選択率 $\sigma$ （テーブル全体の行に対する参照する行の割合）とテーブルに対するアクセスパターンを変えた問合せを実行した。アクセスパターンは、シーケンシャルアクセス（Seq）と、ランダムアクセス（Rand）の2種類を用いた。なおここでのランダムアクセスは、テーブルの全レコードの内ランダムに選択したレコードを、テーブルの先頭から順にアクセスするものとし、選択率が1に近づくほどシーケンシャルアクセスに近いアクセスパターンとなる。

## 5.3 単純ログ適用の基本性能

まず、テストログを用いて単純ログ適用機能を実行した時、すなわちログ適用を単体で行った時の基本性能を測定した。

ログ生成パターンとログ適用手法を変えた時の単純ログ適用の実行時間、発行I/O数、平均I/Oスループットをそれぞれ図5, 図6, 図7に示す。横軸はログ生成パターンを表していて、右のパターンほどランダムアクセスの偏りが大きい。

実行時間を見ると，同一量のログであっても，ランダムアクセスの偏りが大きいほどログ適用にかかる時間が短くなった．また整列ログ適用は，逐次ログ適用と比べて，ログ適用にかかる時間を短縮する効果があり，発行 I/O 数も抑制していることが確認できた．

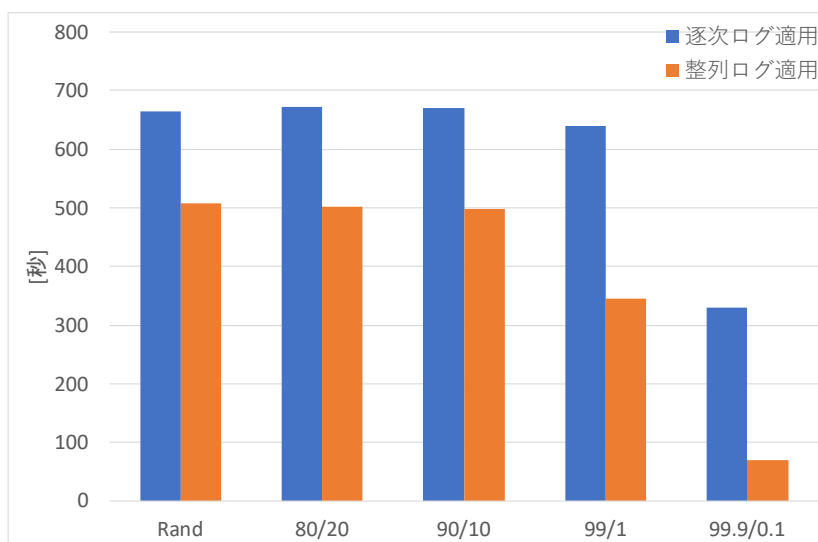


図 5: 単純ログ適用の実行時間

それぞれのログ生成パターン・ログ適用手法における，実行時の毎秒の CPU 利用率と I/O スループットの変動を図 8 から図 17 に示す．CPU 利用率のグラフは，実行プロセスが割り当てられた 1 スレッドのものの種類の内訳を，合計で 100% の積み上げ面グラフとして表している．I/O スループットのグラフは，書き込み (write) を赤線で，読み込み (read) を青線で表しているが，テストログのログ適用においては読み書きは同数となるので，ほぼ重なった線として現れている．

CPU 利用率を見ると，いずれの場合も紫色の %iowait すなわち I/O 待ち時間が大半を占めていて，ログ適用はおおよそ I/O バウンドであることがわかる．I/O スループットのグラフを，例えば Rand・逐次ログ適用の図 8 と Rand・整列ログ適用の図 9 で比較すると，逐次ログ適用に比べて整列ログ適用は I/O スループットが高くなっている．このことと発行 I/O 数が抑制されることが合わせて，整列ログ適用はログ適用にかかる時間を短縮する．



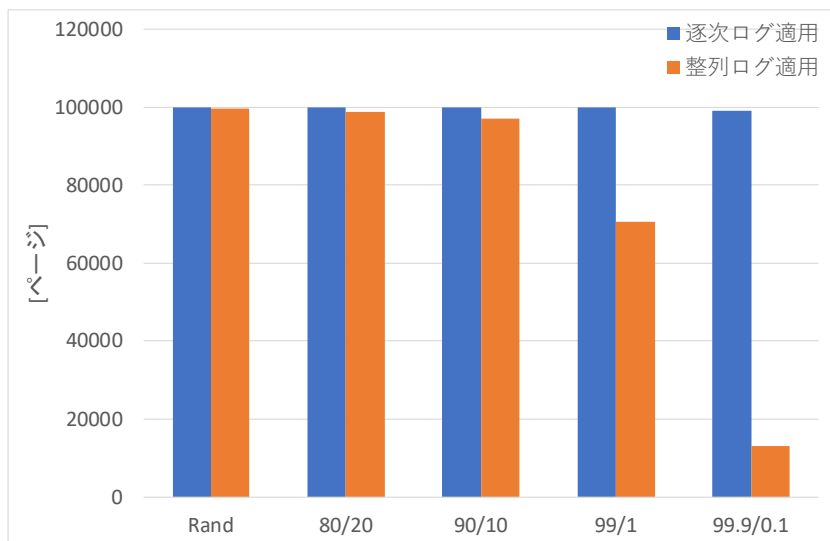


図 6: 単純ログ適用の発行 I/O 数

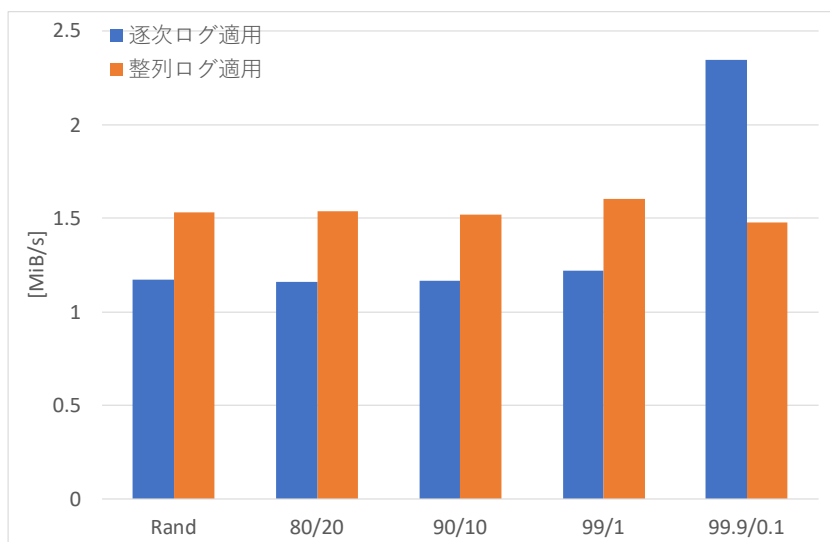


図 7: 単純ログ適用の平均 I/O スループット

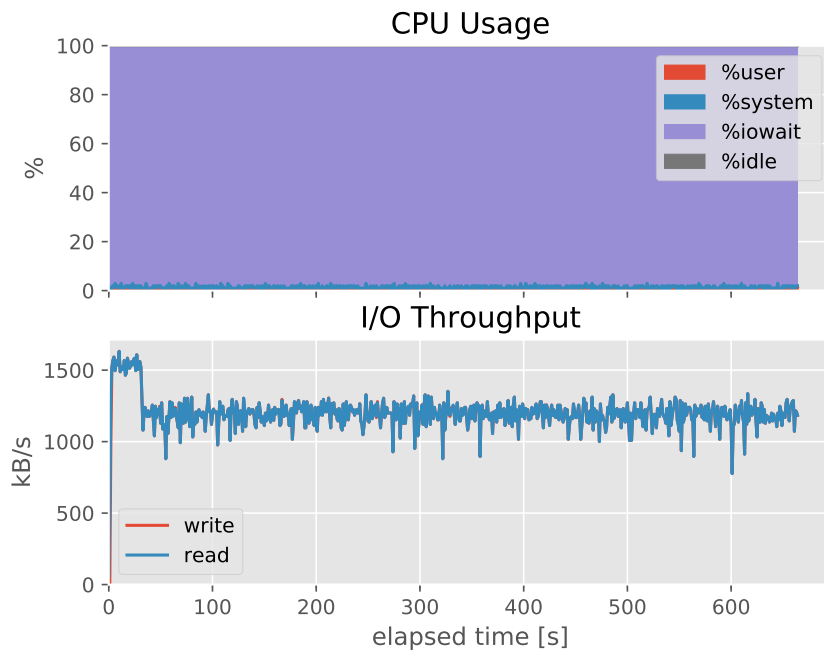


図 8: 単純ログ適用時の毎秒 CPU 利用率・I/O スループット (Rand・逐次ログ適用)

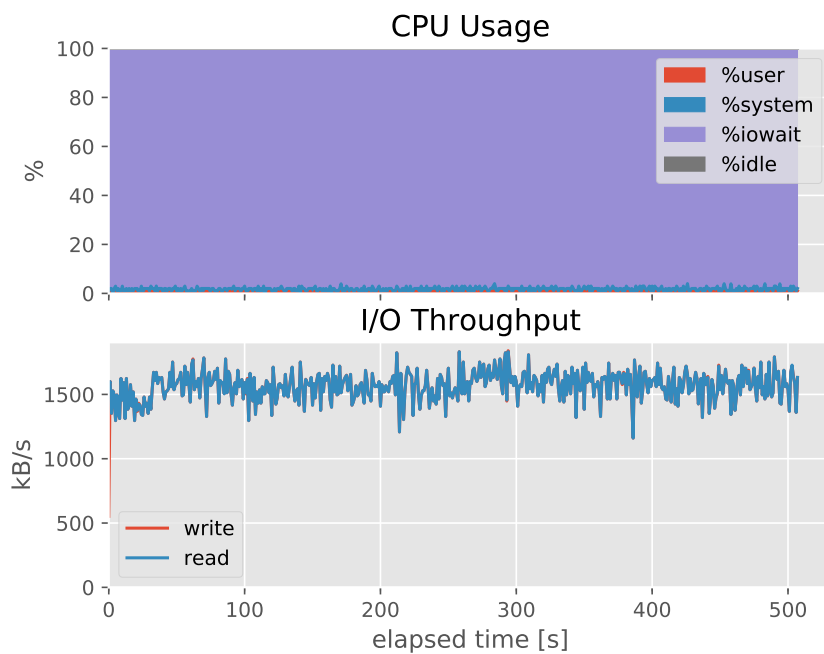


図 9: 単純ログ適用時の毎秒 CPU 利用率・I/O スループット (Rand・整列ログ適用)

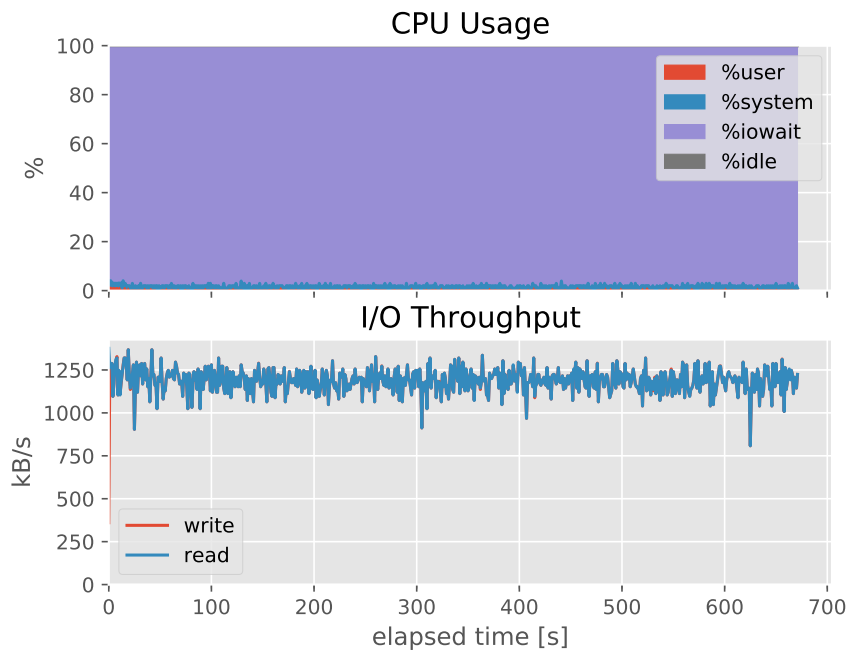


図 10: 単純ログ適用時の毎秒 CPU 利用率・I/O スループット (80/20・逐次ログ適用)

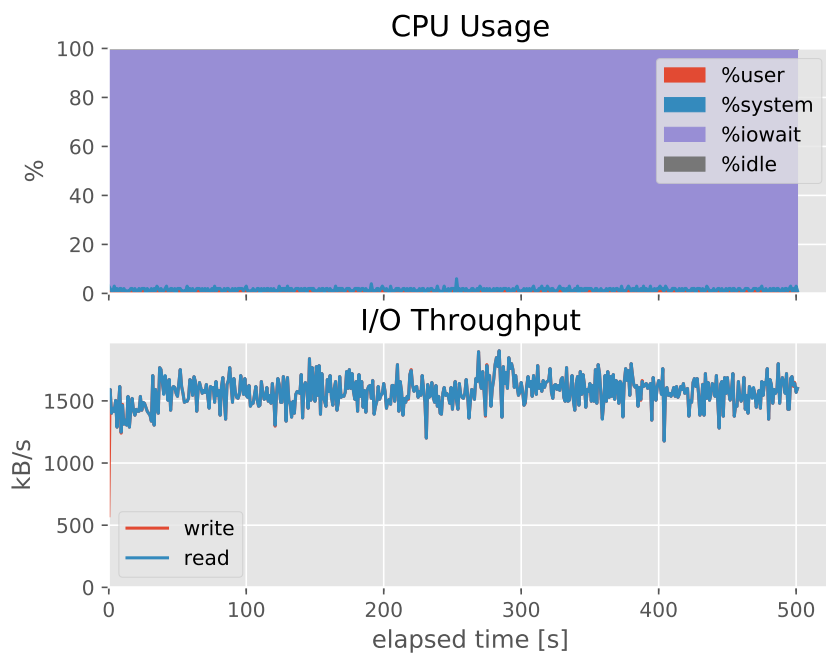


図 11: 単純ログ適用時の毎秒 CPU 利用率・I/O スループット (80/20・整列ログ適用)

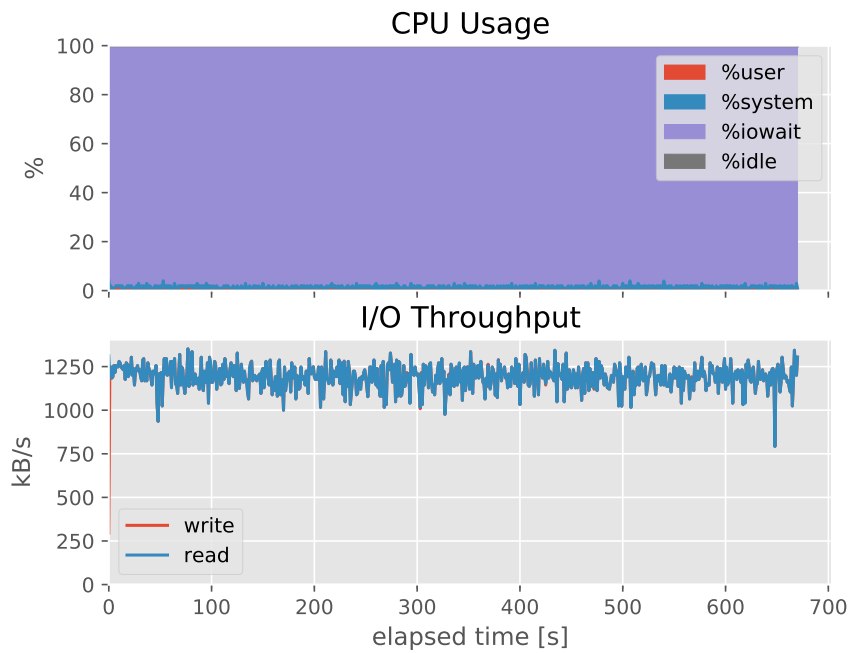


図 12: 単純ログ適用時の毎秒 CPU 利用率・I/O スループット (90/10・逐次ログ適用)

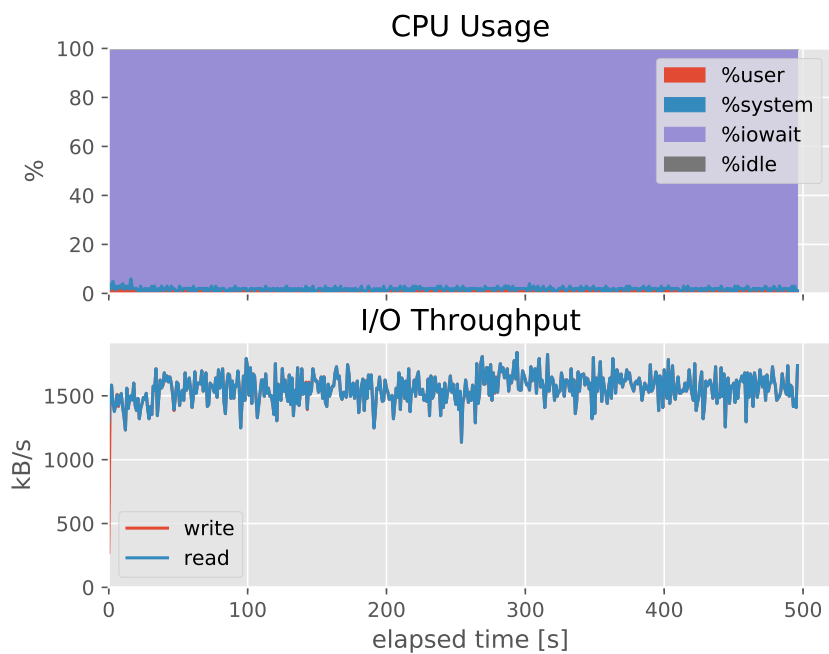


図 13: 単純ログ適用時の毎秒 CPU 利用率・I/O スループット (90/10・整列ログ適用)

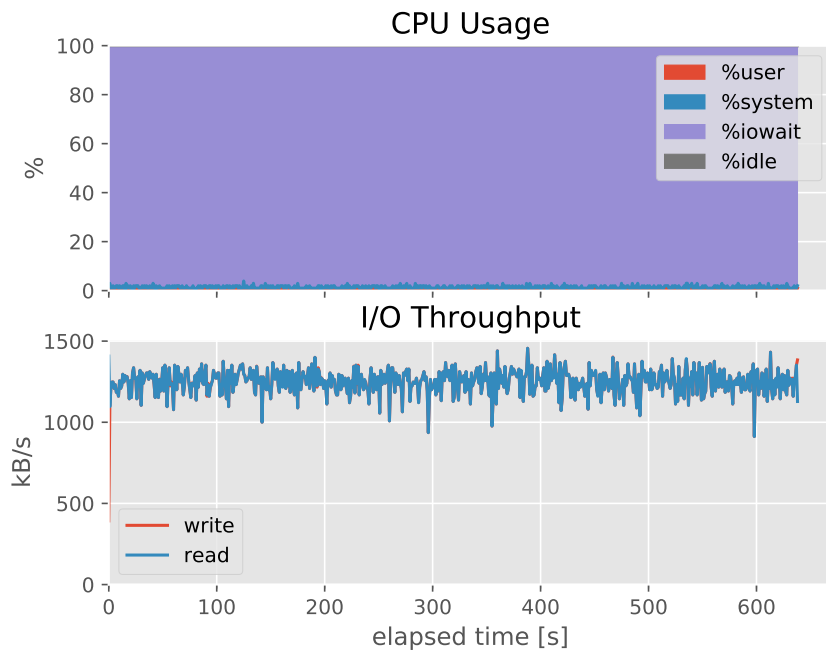


図 14: 単純ログ適用時の毎秒 CPU 利用率・I/O スループット (99/1・逐次ログ適用)

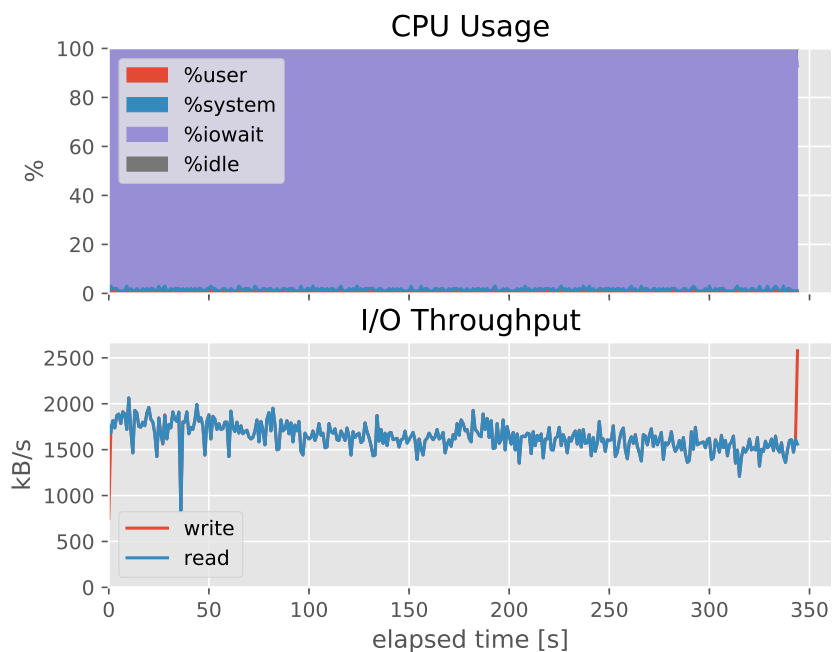


図 15: 単純ログ適用時の毎秒 CPU 利用率・I/O スループット (99/1・整列ログ適用)

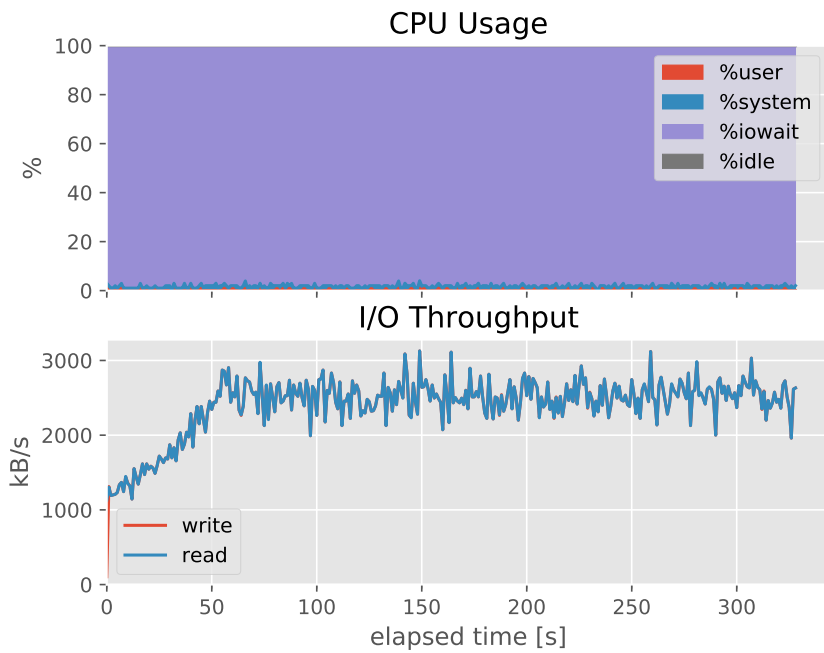


図 16: 単純ログ適用時の毎秒 CPU 利用率・I/O スループット (99.9/0.1・逐次ログ適用)

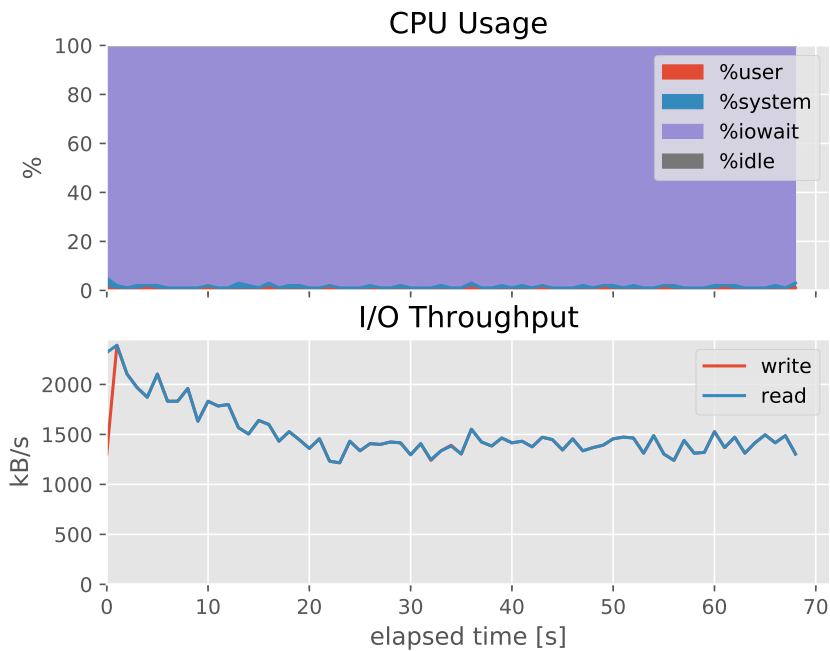


図 17: 単純ログ適用時の毎秒 CPU 利用率・I/O スループット (99.9/0.1・整列ログ適用)

## 5.4 単純問合せ実行の基本性能

次に、様々な選択率の単純問合せ処理を実行した時の実行時間、発行I/O量、平均I/Oスループットをそれぞれ図18、図19、図20に示す。これらのグラフは両対数プロットになっている。

選択率とは、対象となるデータベースのテーブルの全レコードの内、どの程度のレコードにアクセスするかの割合を表し、例えば選択率1の場合はテーブルの全レコードにアクセスし、選択率 $1e-2$ つまり0.01の場合はテーブルの1%のレコードにアクセスするということを意味する。グラフ中で青色の点 (Seq) はシーケンシャルアクセス問合せを表していて、オレンジ色の点 (Rand) はランダムアクセス問合せを表している。

同じ選択率であってもディスクアクセスのパターンが異なるので実行時間が大きく異なる。青色のシーケンシャルアクセス問合せは、実行時間がほぼ選択率に比例している。対してオレンジ色のランダムアクセス問合せは、選択率が低いときには同様に実行時間がほぼ選択率に比例しているが、選択率が $1e-5$ 以上において比例関係との乖離が見られた。これは発行I/Oのシーク距離が短くなりI/O当たりの必要時間が短くなったためと考えられる。

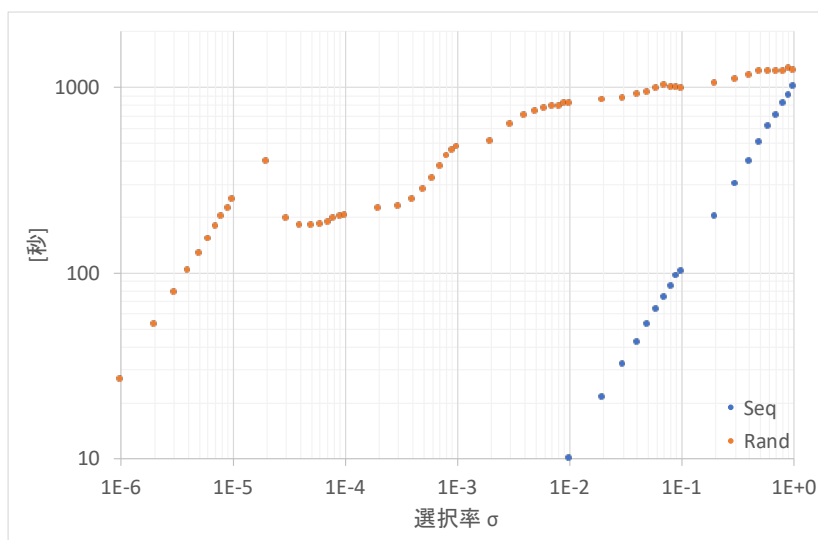


図 18: 単純問合せ実行の実行時間

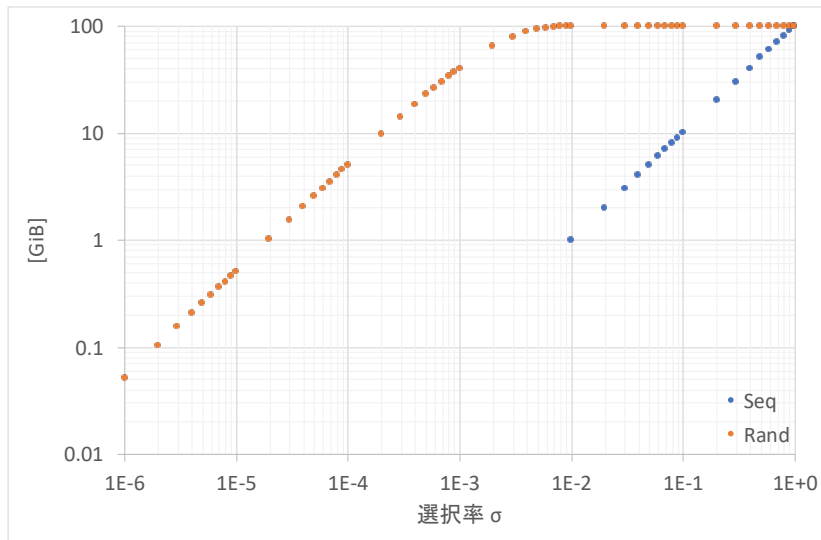


図 19: 単純問合せ実行の発行 I/O 量

また、シーケンシャル問合せにおける選択率別の毎秒の CPU 利用率と I/O スループットを図 21 から図 30 に、ランダムアクセス問合せにおける選択率別の毎秒の CPU 利用率と I/O スループットを図 31 から図 85 に示す。

ランダムアクセス問合せの特徴としては、例えば図 40 の選択率  $1e-5$  の場合を見ると、ほとんど CPU 時間を消費せず I/O バウンドなタスクとなっている。他方、シーケンシャルアクセス問合せの特徴としては、例えば図 21 の選択率 0.1 の場合を見ると、より CPU 時間を消費する I/O バウンドなタスクになっている。



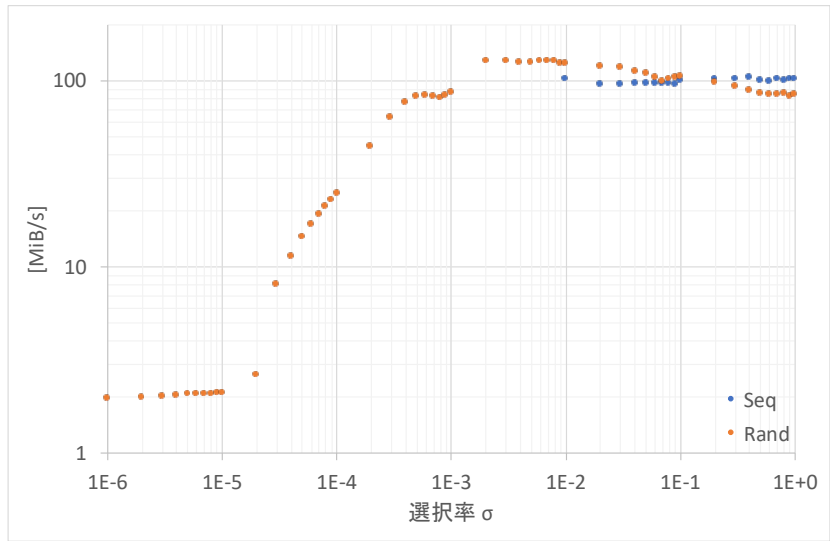


図 20: 単純問合せ実行の平均 I/O スループット

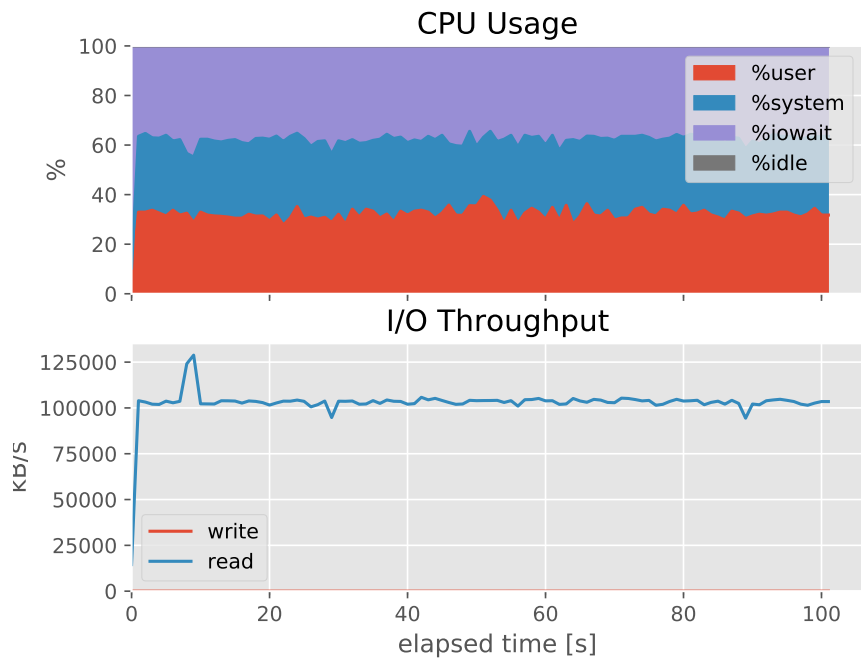


図 21: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Seq・ $\sigma = 0.1$ )

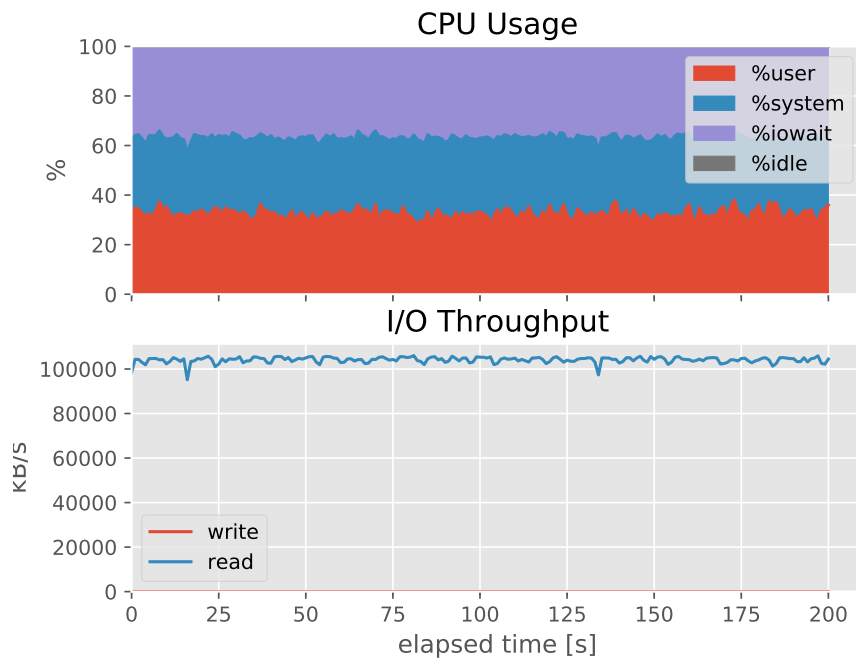


図 22: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Seq ·  $\sigma = 0.2$ )

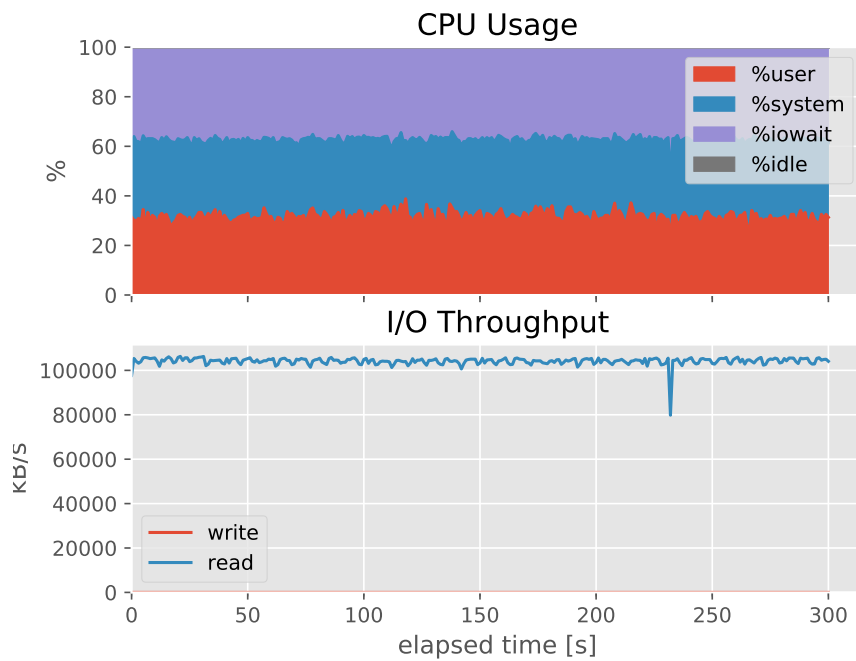


図 23: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Seq ·  $\sigma = 0.3$ )

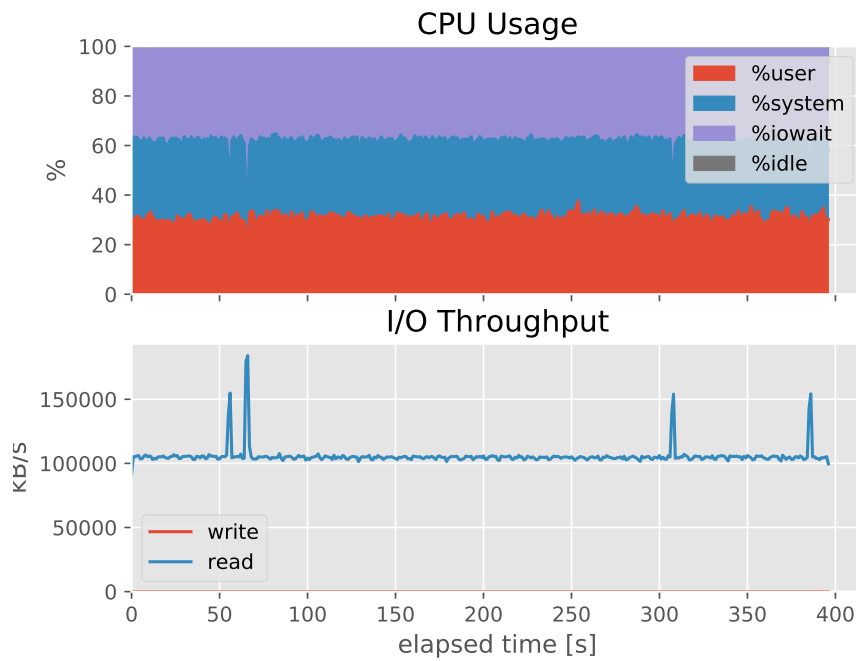


図 24: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Seq ·  $\sigma = 0.4$ )

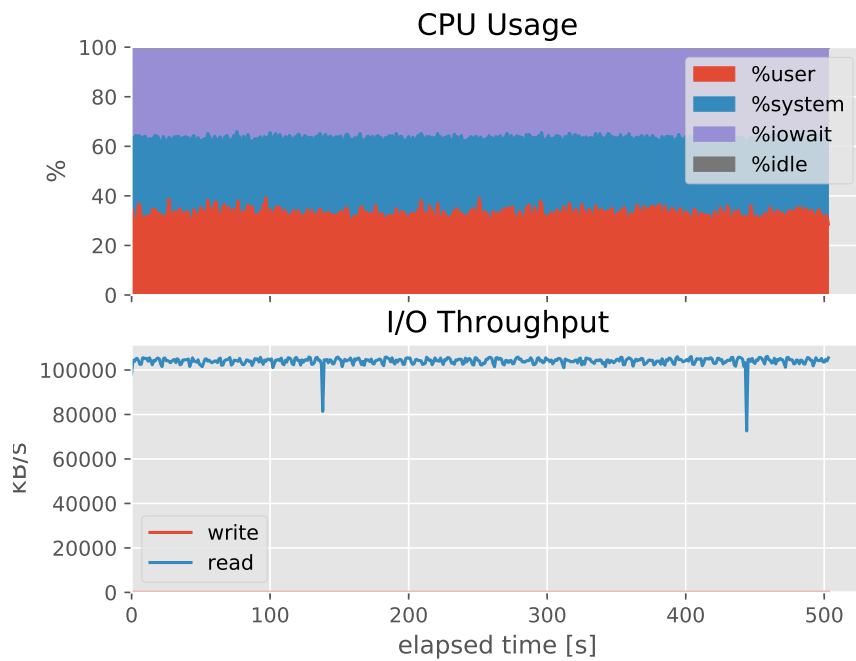


図 25: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Seq ·  $\sigma = 0.5$ )

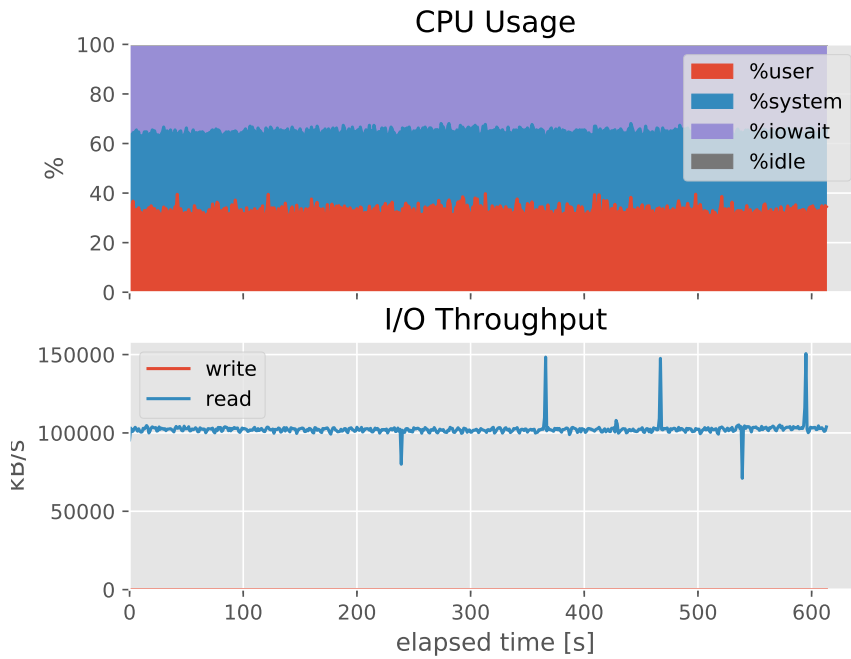


図 26: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Seq ·  $\sigma = 0.6$ )

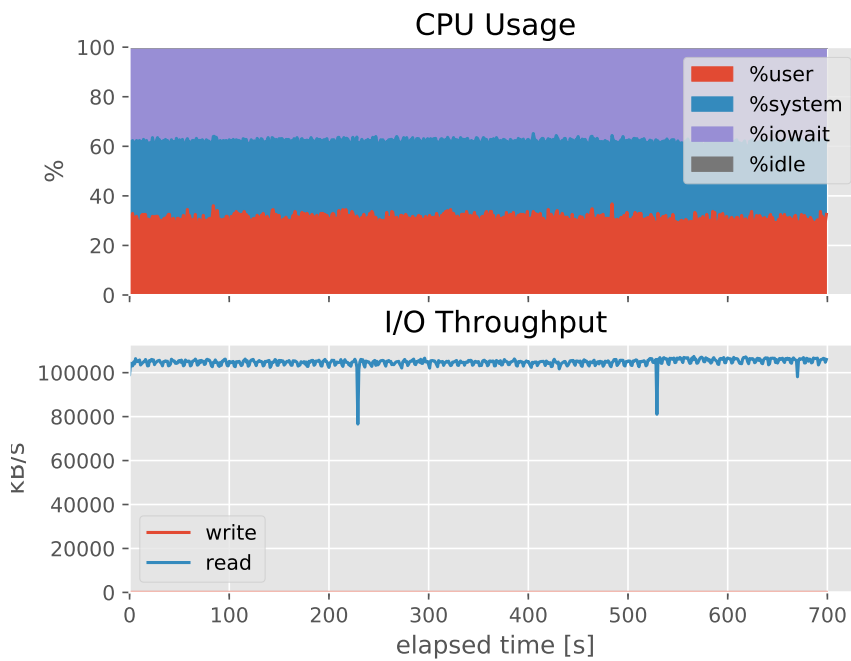


図 27: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Seq ·  $\sigma = 0.7$ )

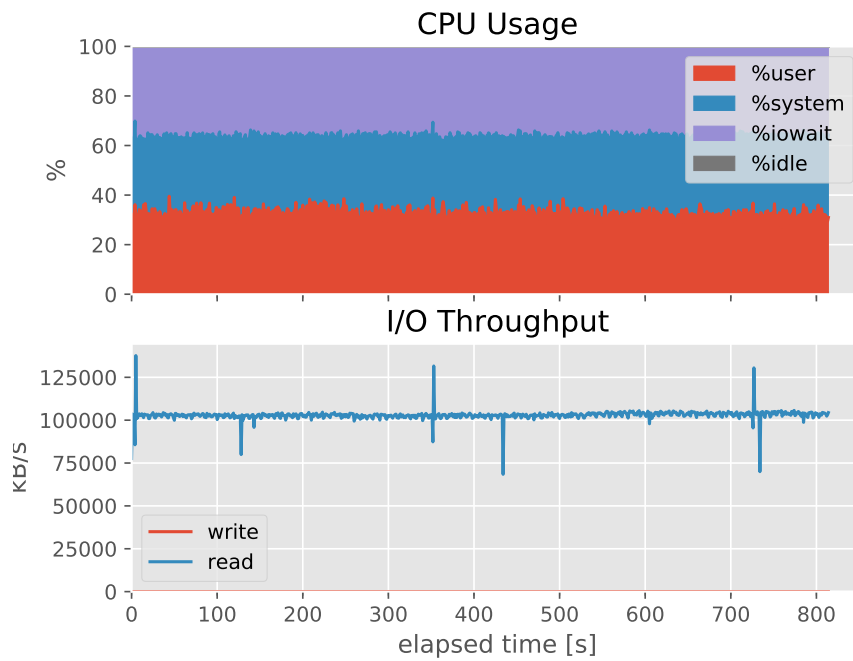


図 28: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Seq ·  $\sigma = 0.8$ )

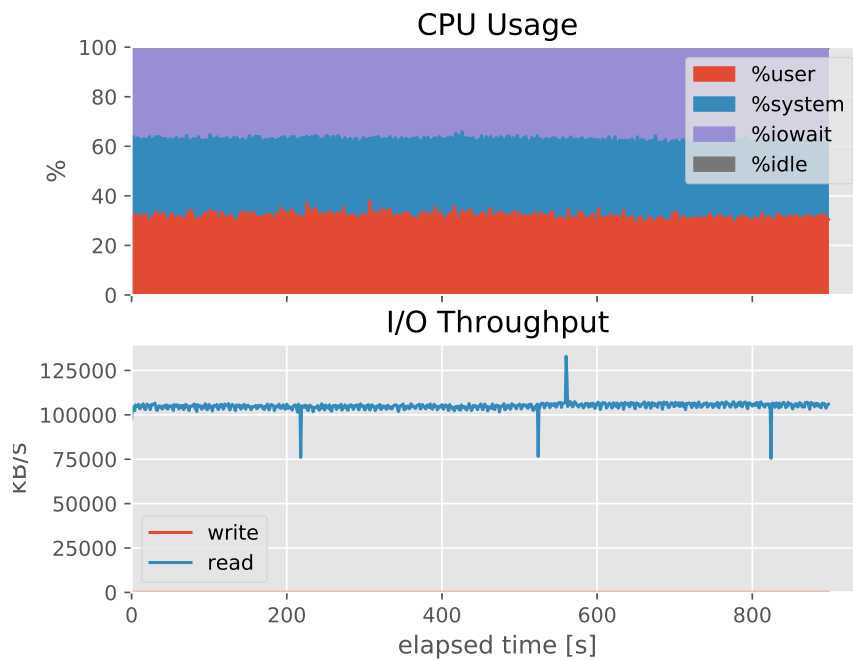


図 29: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Seq ·  $\sigma = 0.9$ )

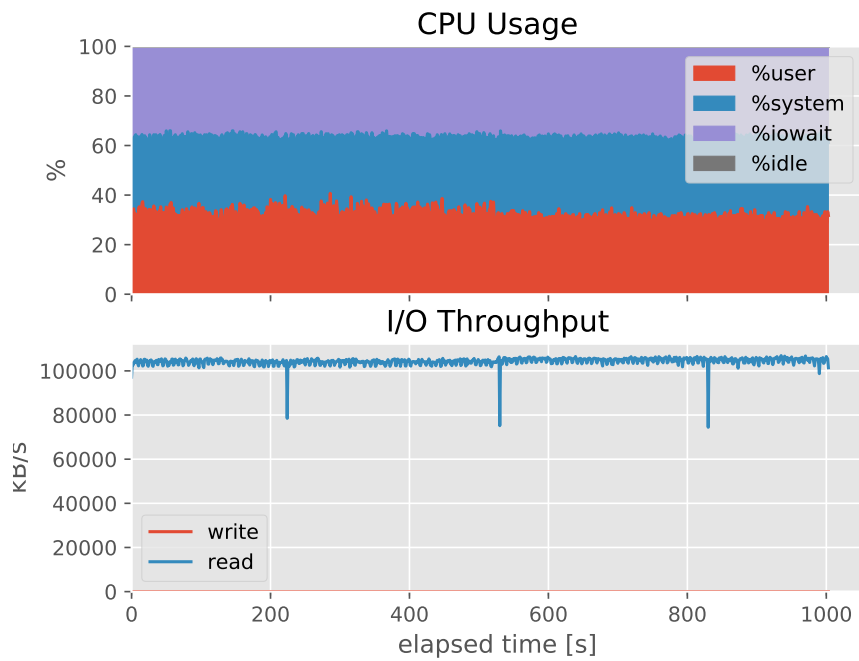


図 30: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Seq ·  $\sigma = 1$ )

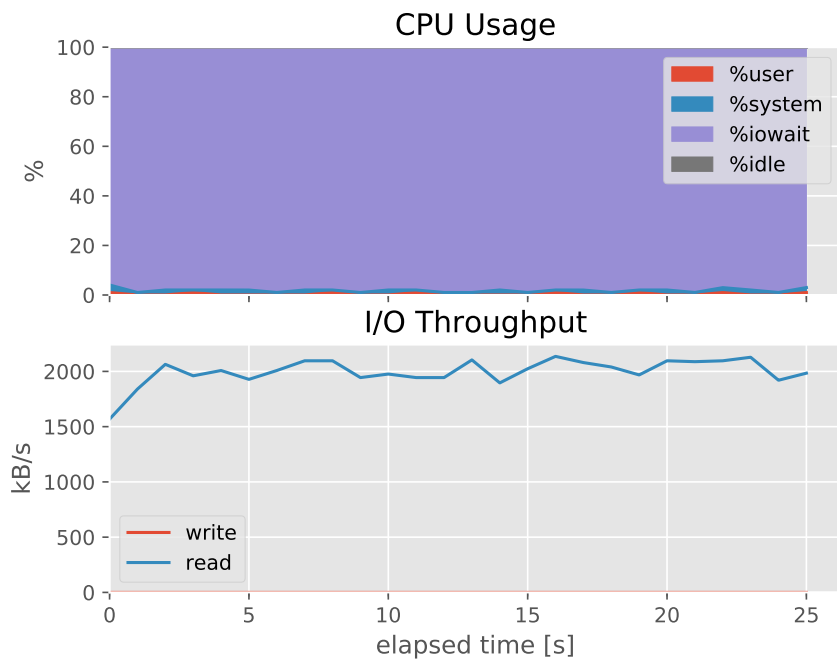


図 31: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand ·  $\sigma = 1e-6$ )

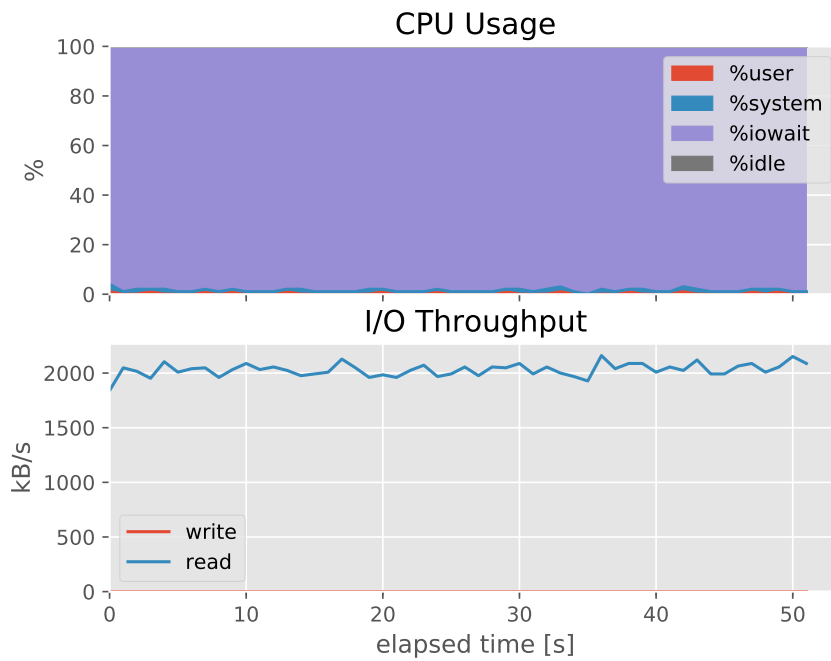


図 32: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット ( $\text{Rand} \cdot \sigma = 2e-6$ )

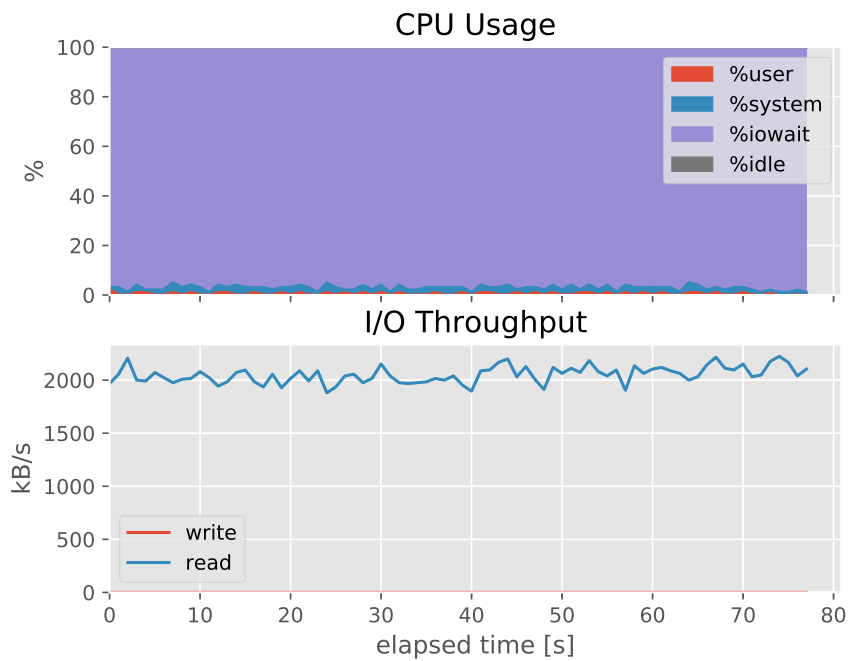


図 33: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット ( $\text{Rand} \cdot \sigma = 3e-6$ )

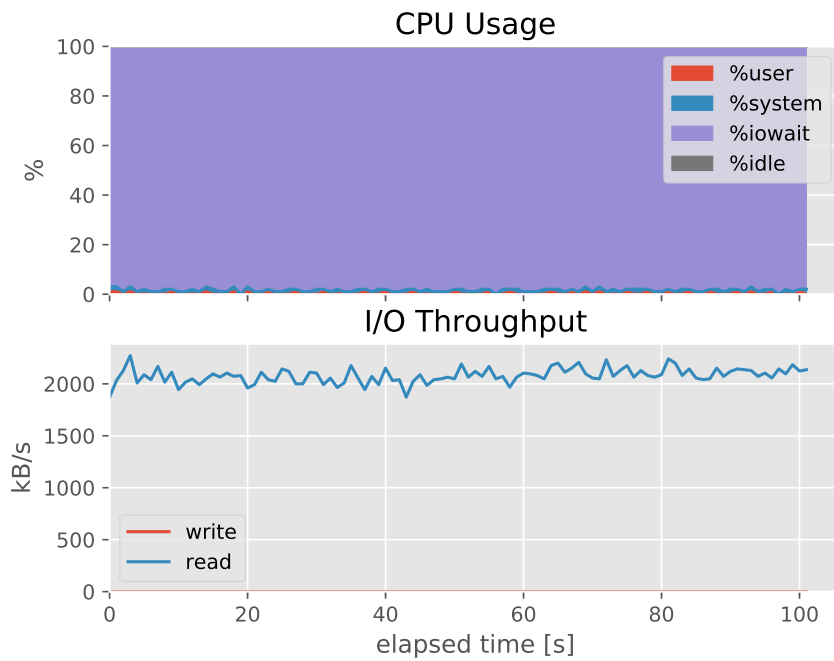


図 34: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット ( $\text{Rand} \cdot \sigma = 4e-6$ )

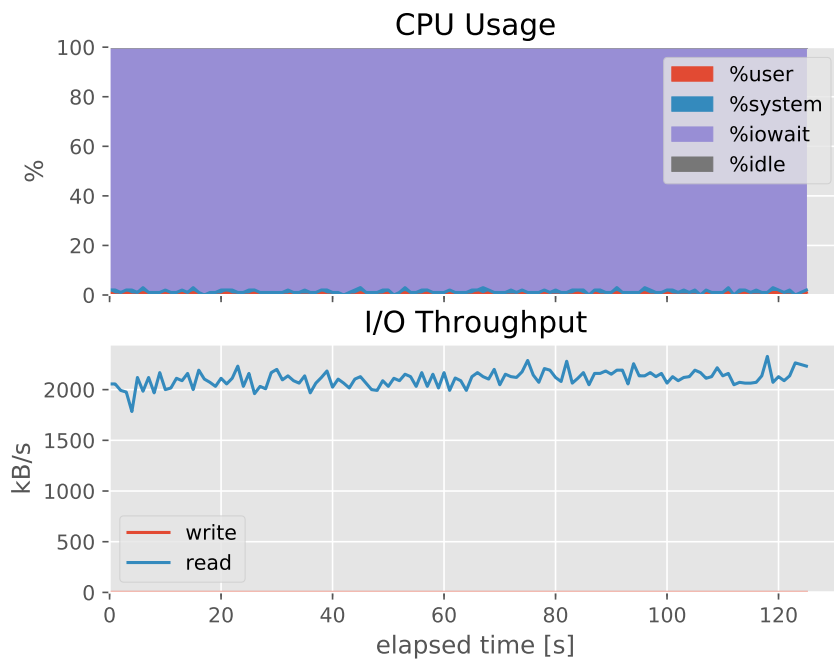


図 35: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット ( $\text{Rand} \cdot \sigma = 5e-6$ )



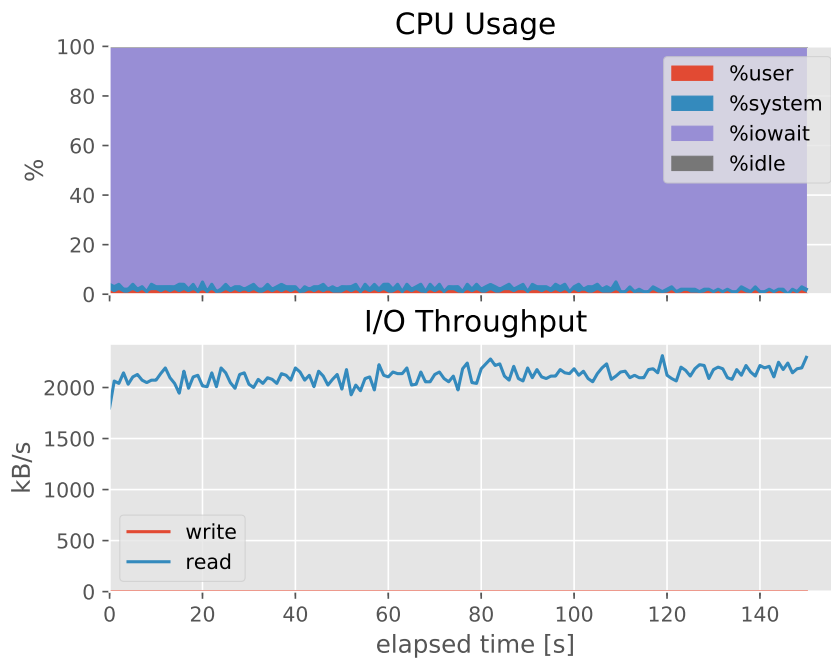


図 36: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット ( $\text{Rand} \cdot \sigma = 6e-6$ )

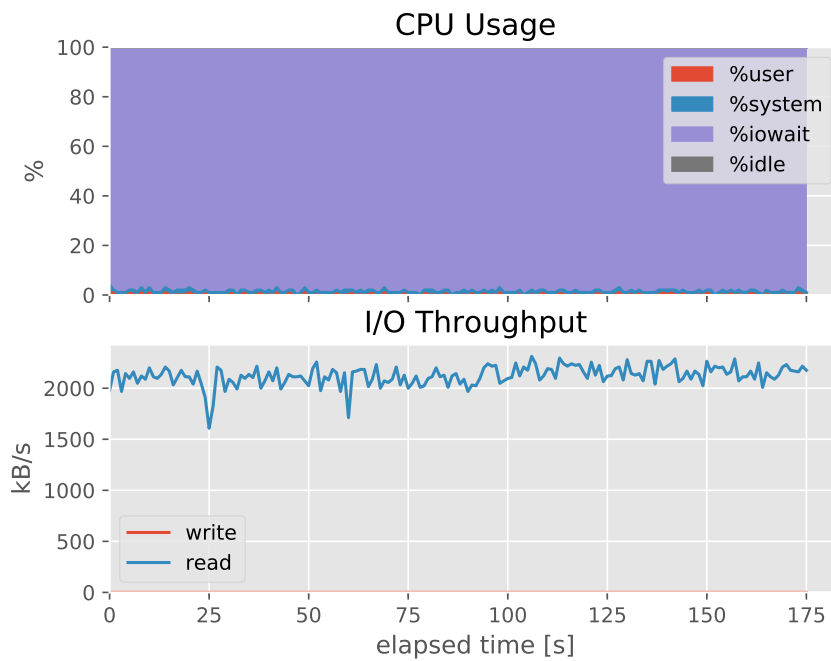


図 37: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット ( $\text{Rand} \cdot \sigma = 7e-6$ )

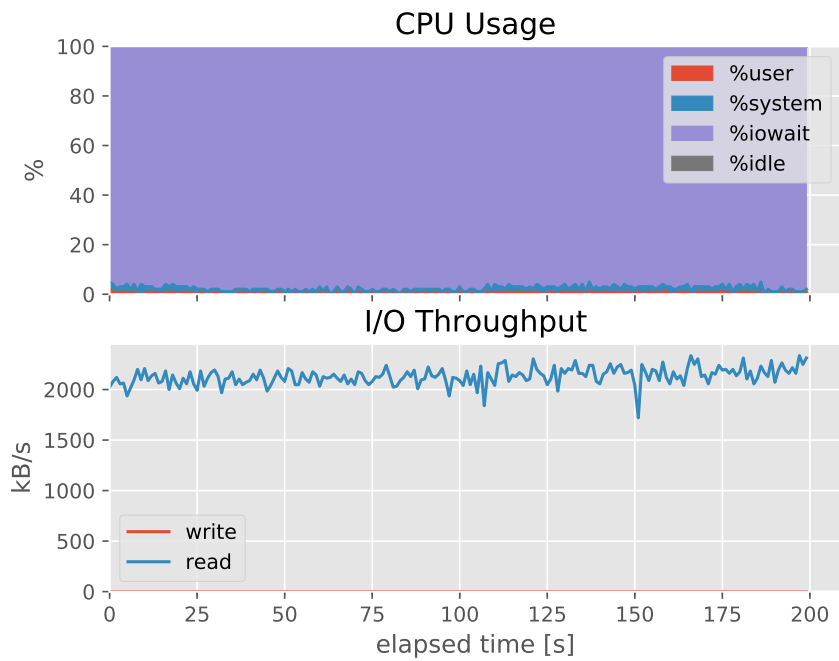


図 38: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット ( $\text{Rand} \cdot \sigma = 8e-6$ )

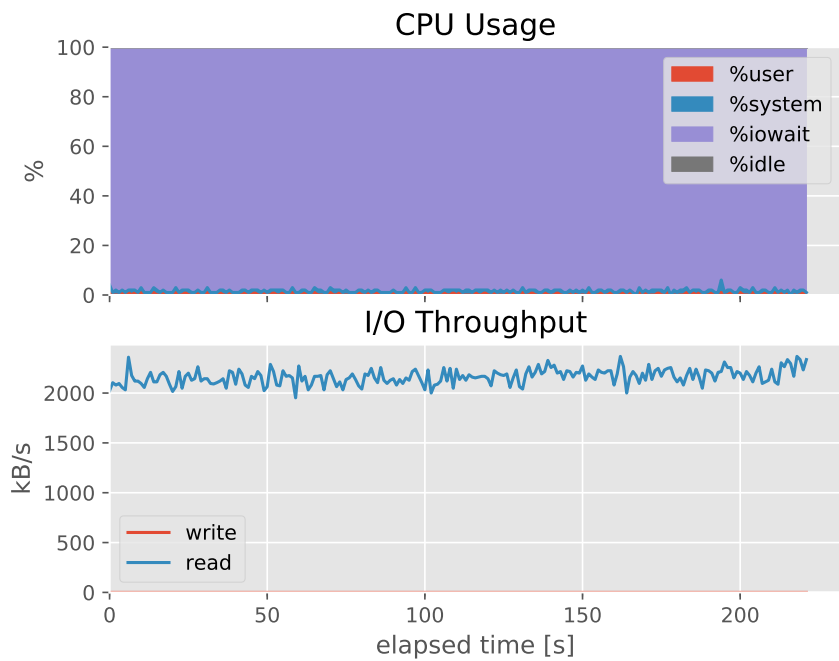


図 39: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット ( $\text{Rand} \cdot \sigma = 9e-6$ )

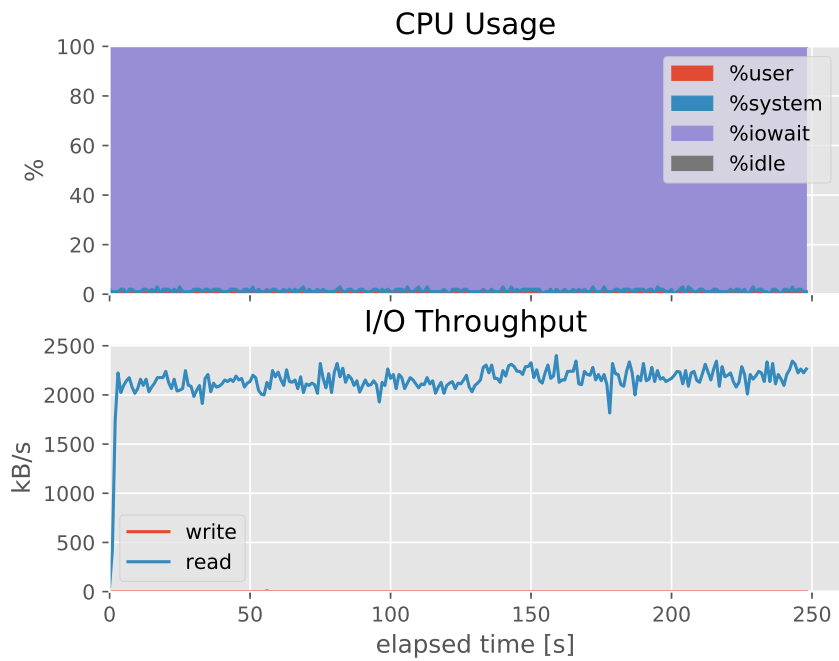


図 40: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット ( $\text{Rand} \cdot \sigma = 1e-5$ )

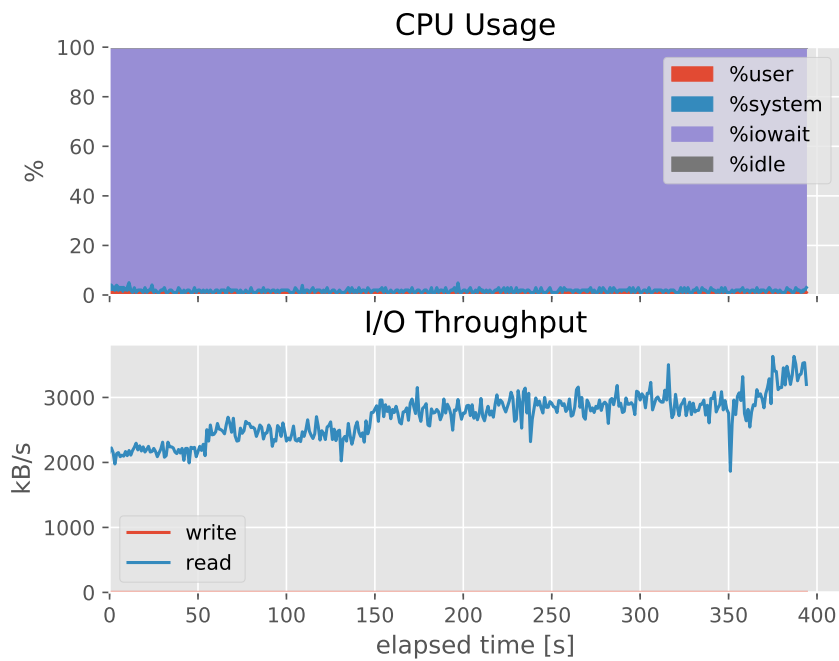


図 41: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット ( $\text{Rand} \cdot \sigma = 2e-5$ )

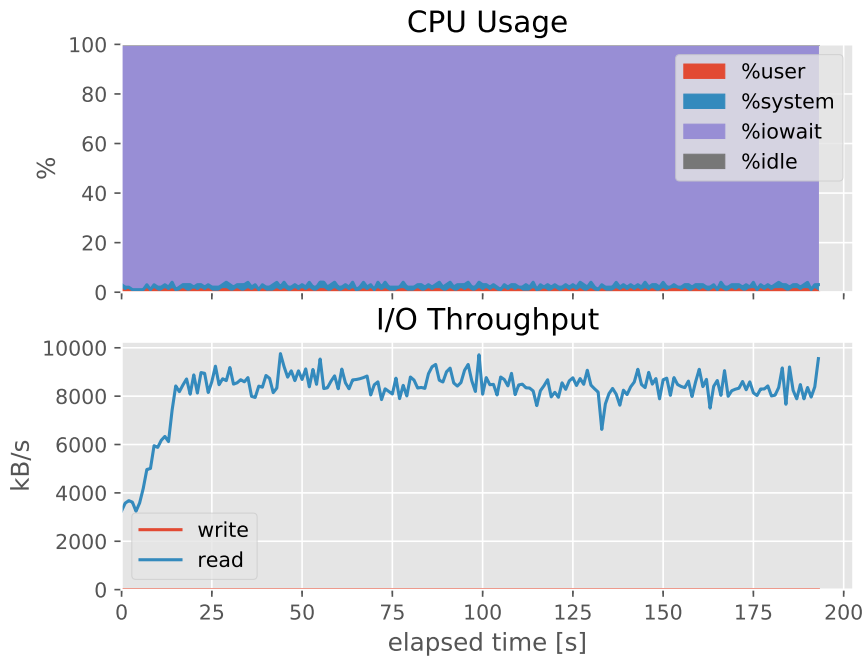


図 42: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット ( $\text{Rand} \cdot \sigma = 3e-5$ )

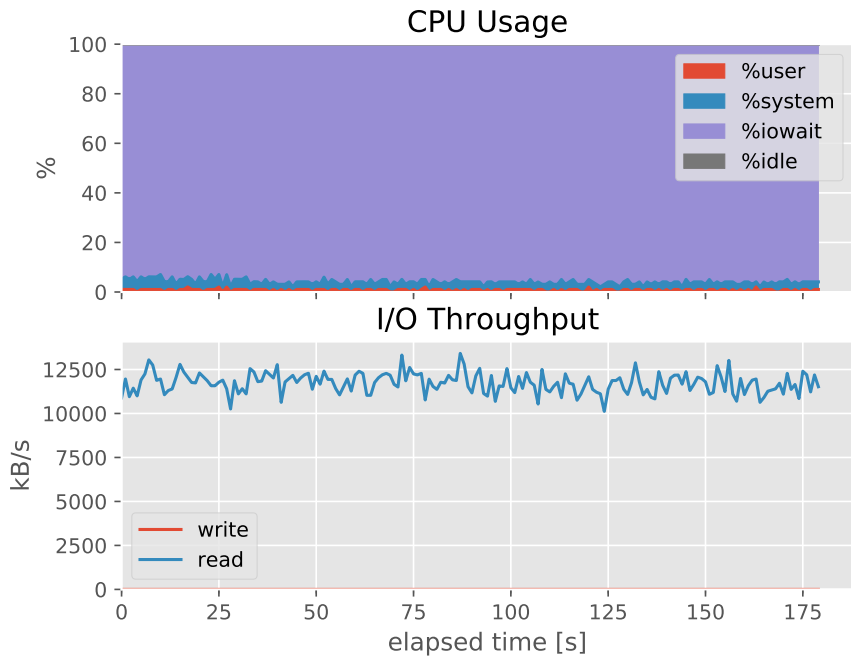


図 43: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット ( $\text{Rand} \cdot \sigma = 4e-5$ )

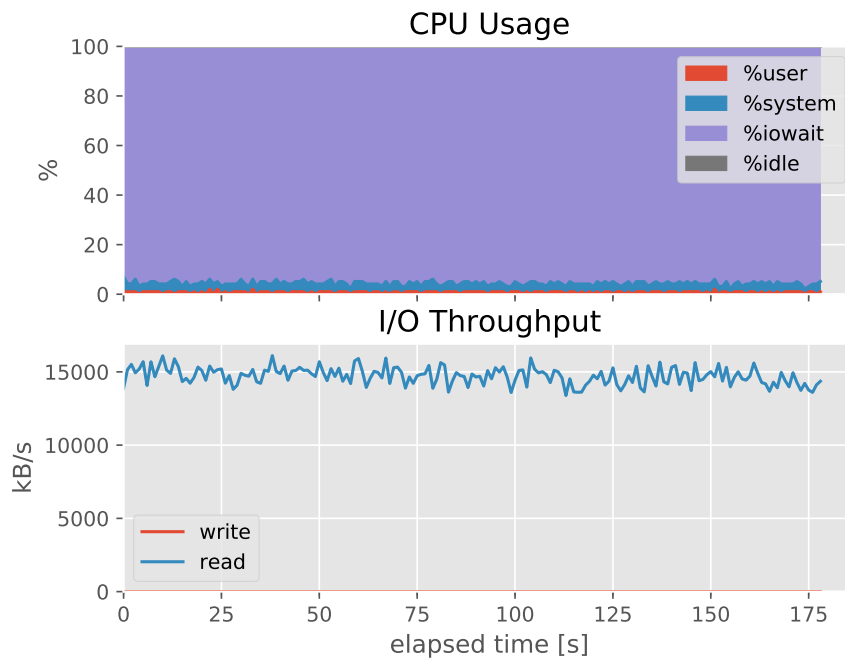


図 44: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット ( $\text{Rand} \cdot \sigma = 5e-5$ )

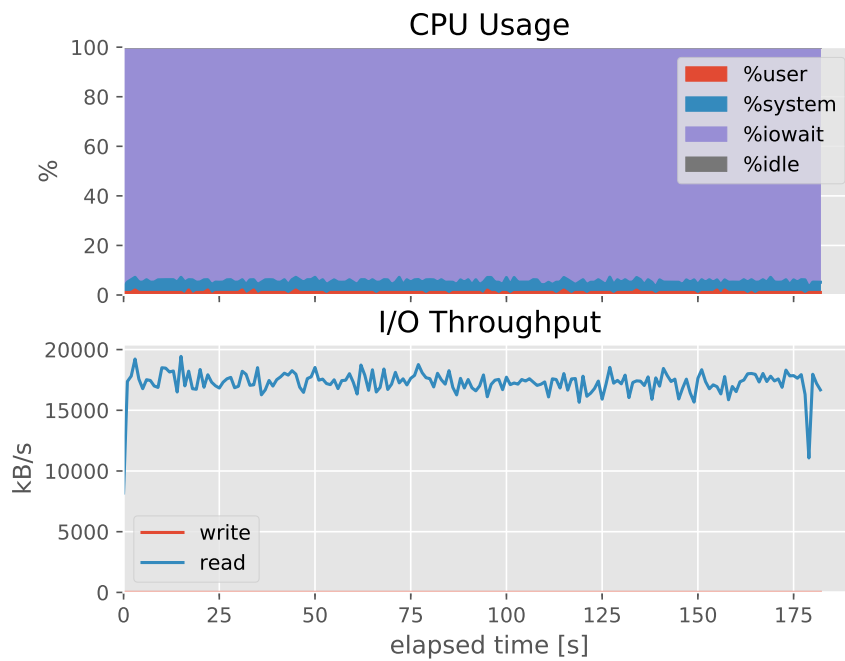


図 45: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット ( $\text{Rand} \cdot \sigma = 6e-5$ )

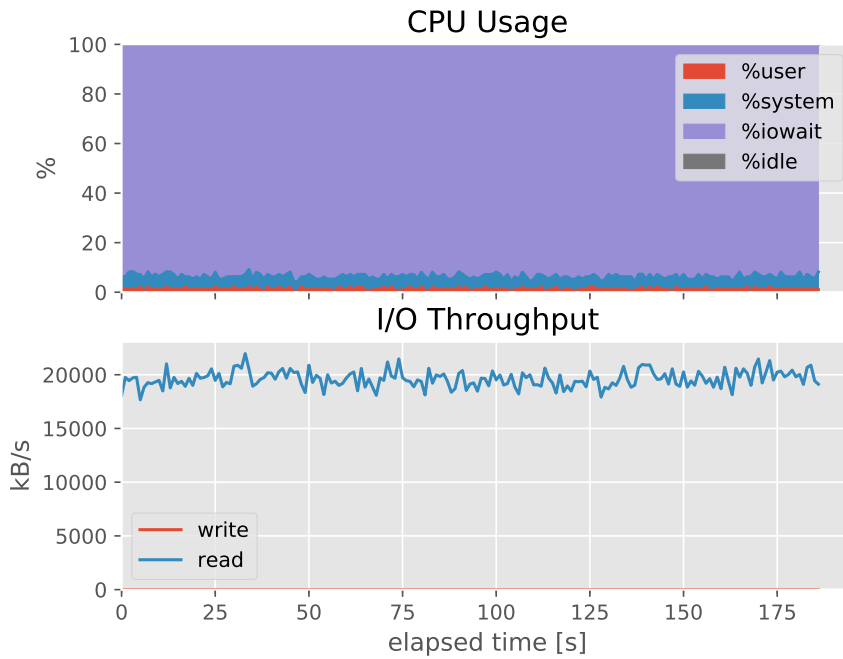


図 46: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット ( $\text{Rand} \cdot \sigma = 7e-5$ )

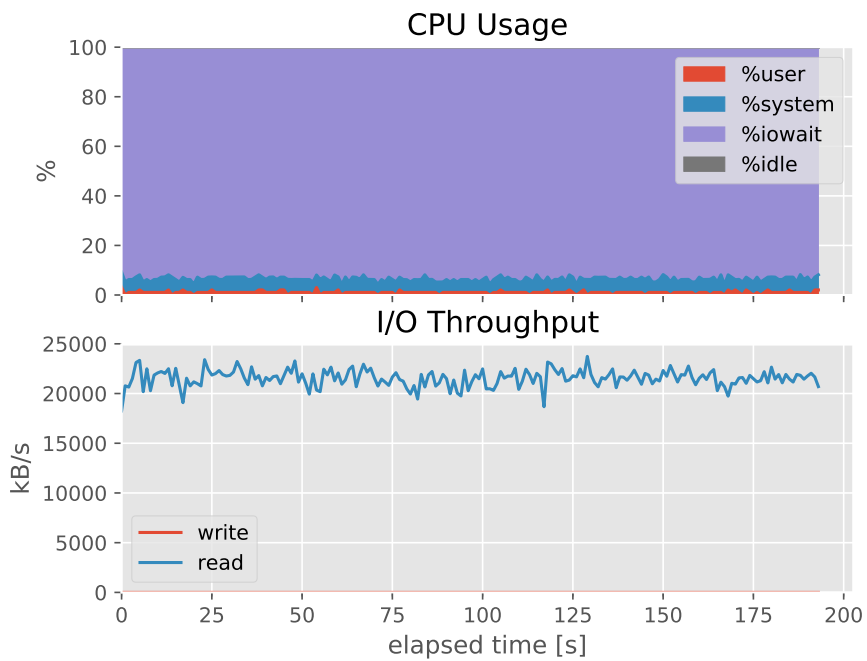


図 47: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット ( $\text{Rand} \cdot \sigma = 8e-5$ )

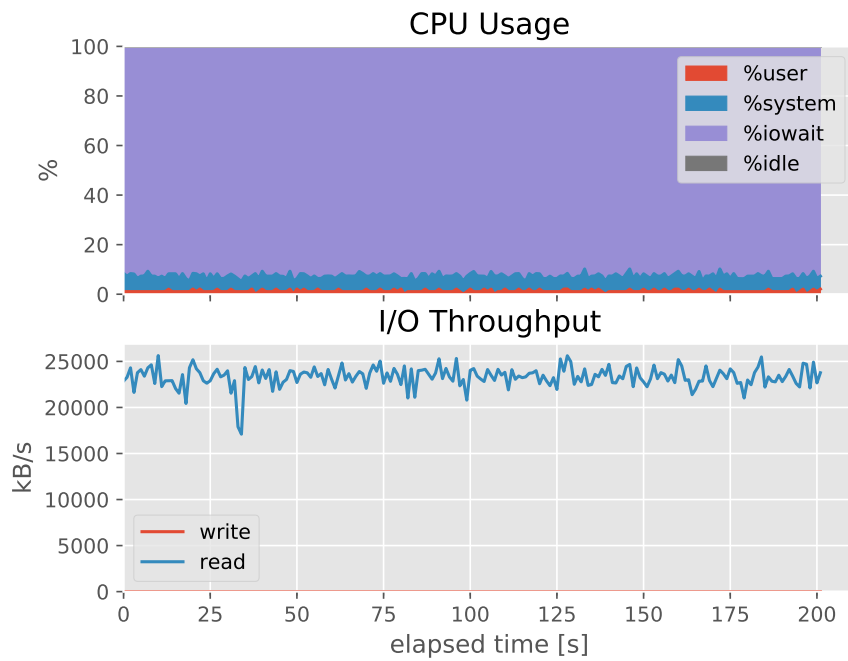


図 48: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット ( $\text{Rand} \cdot \sigma = 9e-5$ )

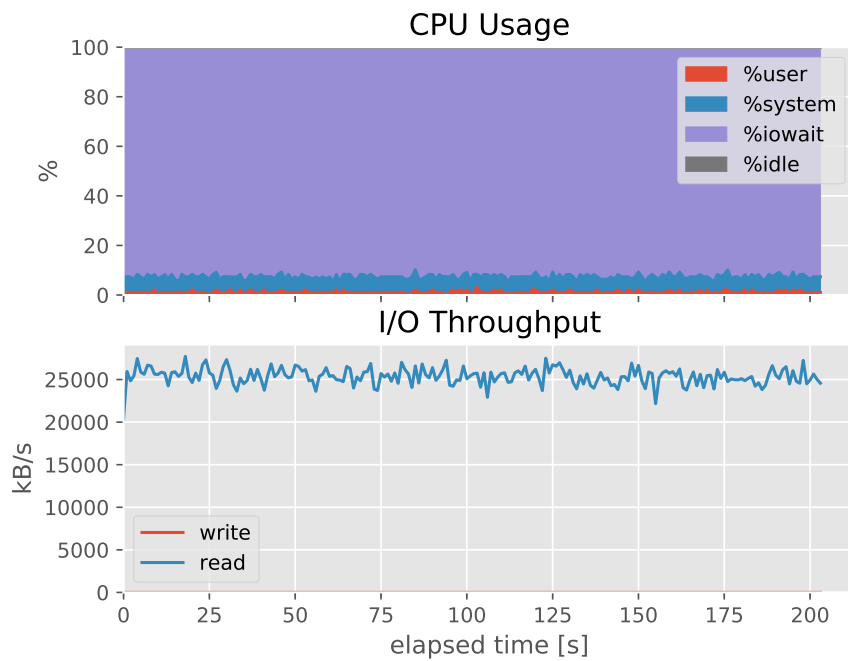


図 49: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット ( $\text{Rand} \cdot \sigma = 1e-4$ )

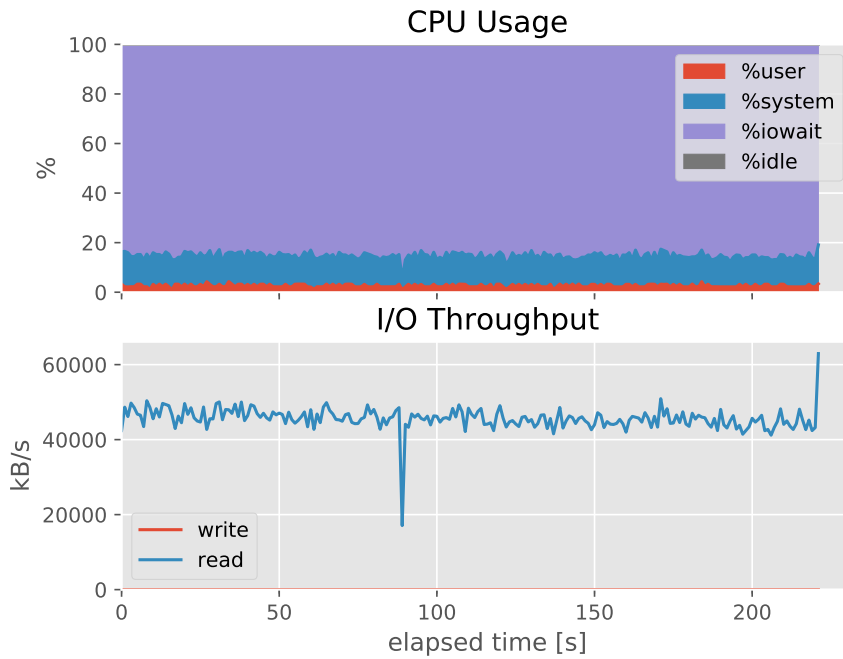


図 50: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット ( $\text{Rand} \cdot \sigma = 2e-4$ )

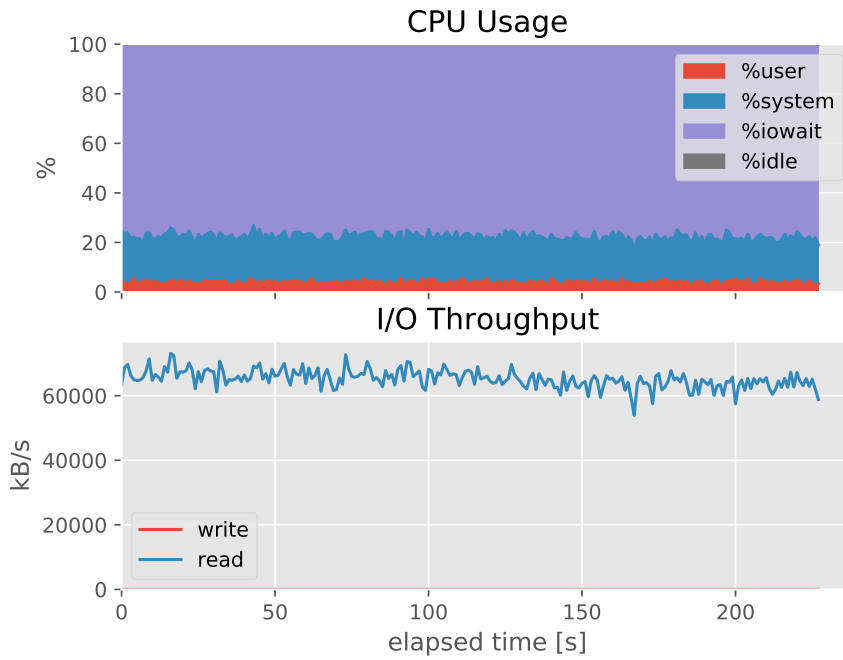


図 51: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット ( $\text{Rand} \cdot \sigma = 3e-4$ )



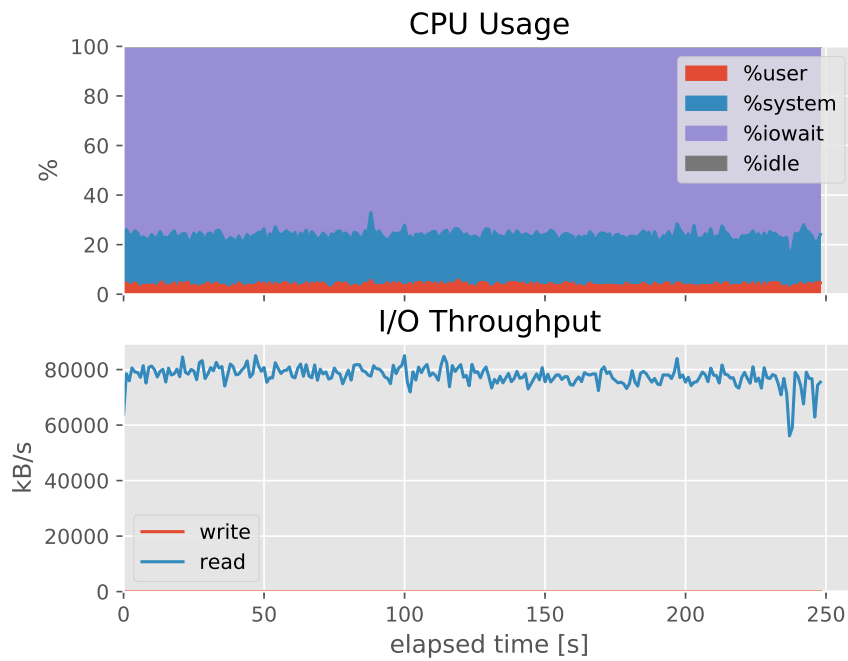


図 52: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット ( $\text{Rand} \cdot \sigma = 4e-4$ )

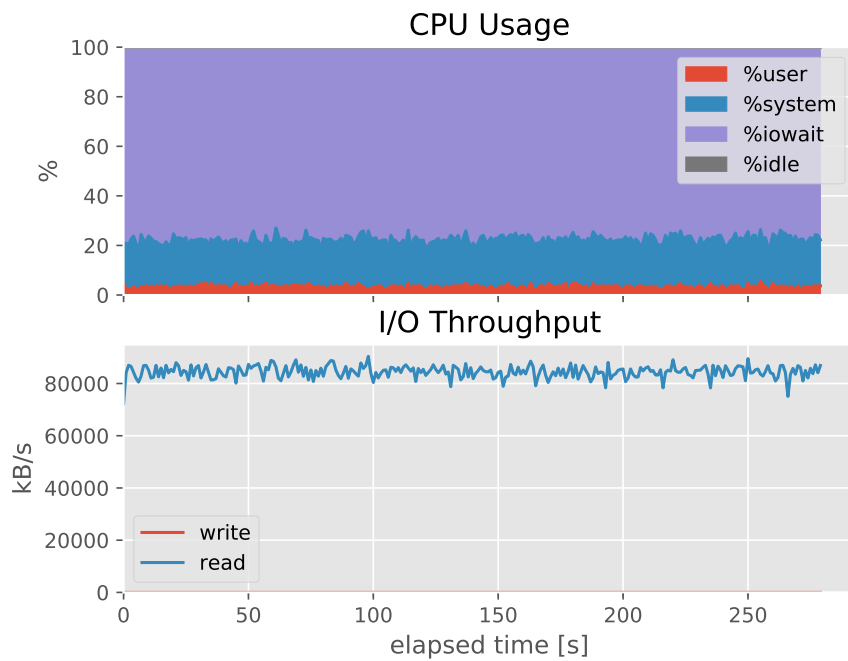


図 53: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット ( $\text{Rand} \cdot \sigma = 5e-4$ )

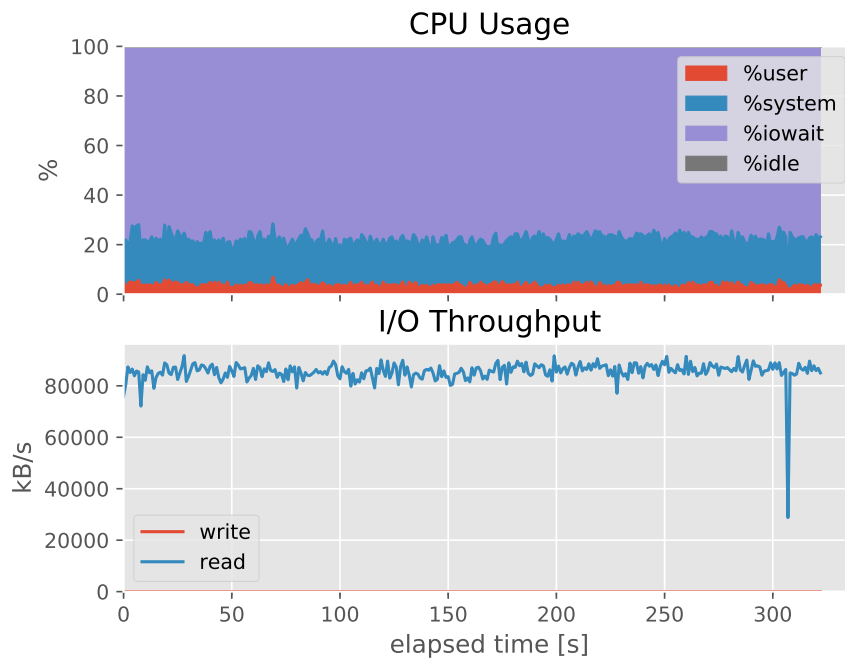


図 54: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット ( $\text{Rand} \cdot \sigma = 6e-4$ )

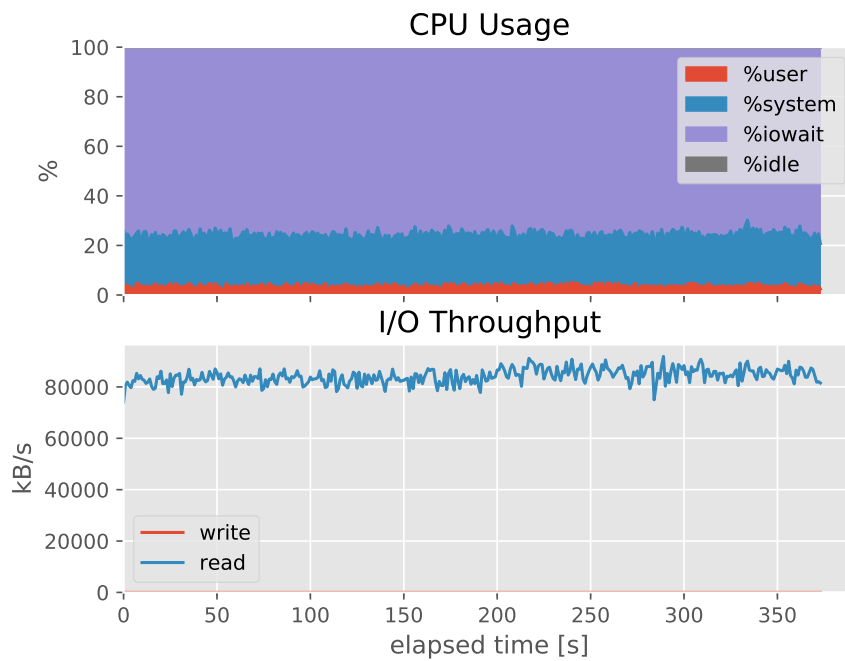


図 55: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット ( $\text{Rand} \cdot \sigma = 7e-4$ )

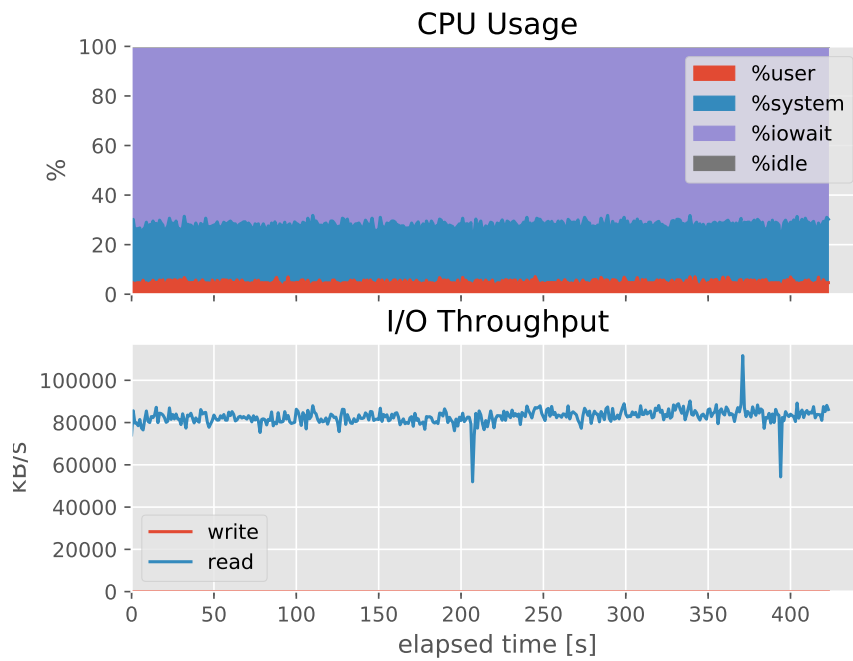


図 56: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット ( $\text{Rand} \cdot \sigma = 8e-4$ )

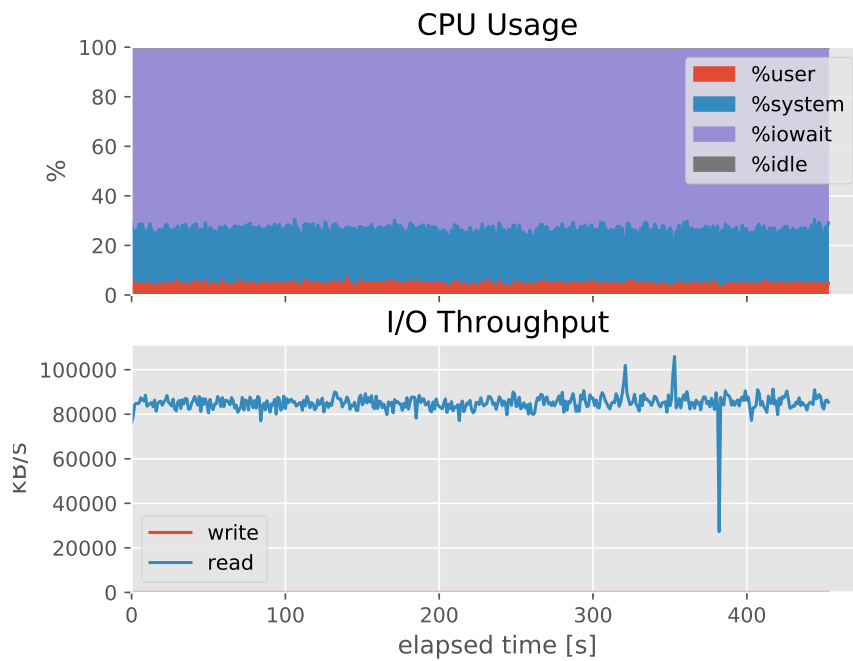


図 57: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット ( $\text{Rand} \cdot \sigma = 9e-4$ )

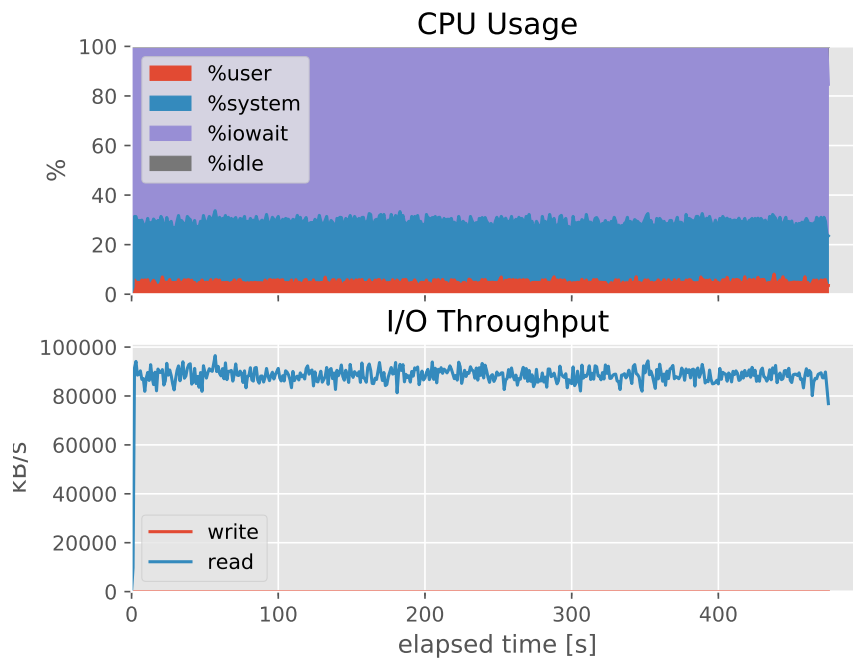


図 58: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット ( $\text{Rand} \cdot \sigma = 1e-3$ )

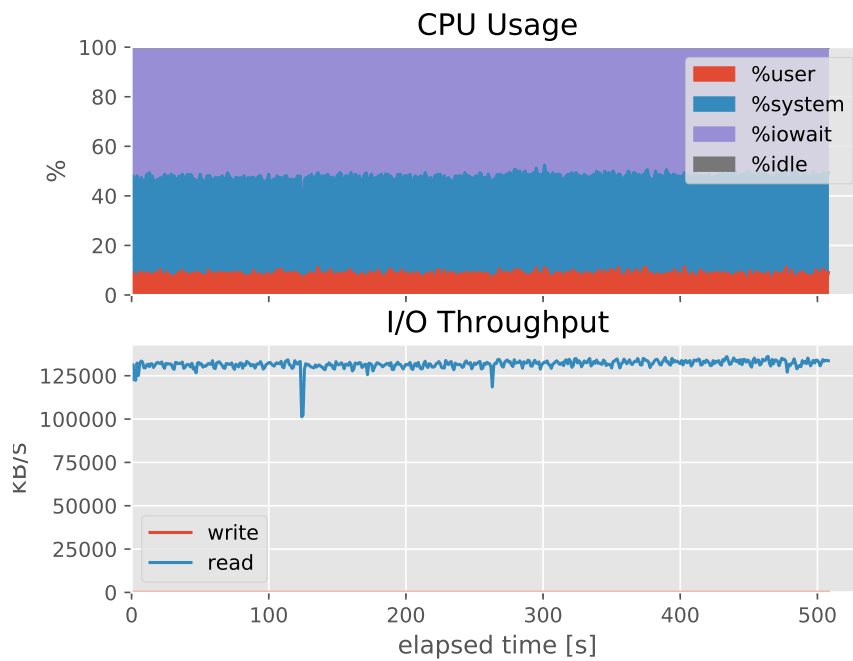


図 59: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット ( $\text{Rand} \cdot \sigma = 2e-3$ )

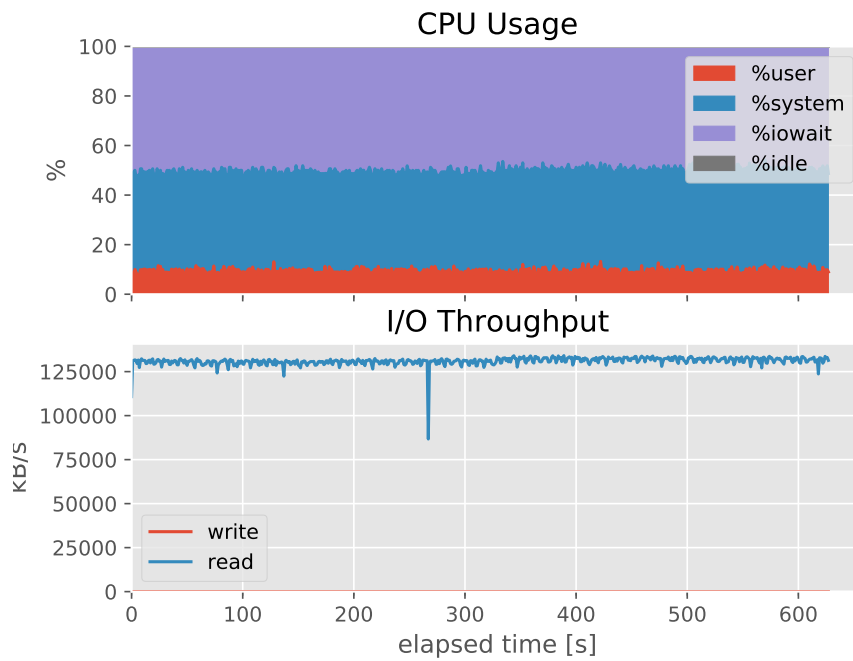


図 60: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット ( $\text{Rand} \cdot \sigma = 3e-3$ )

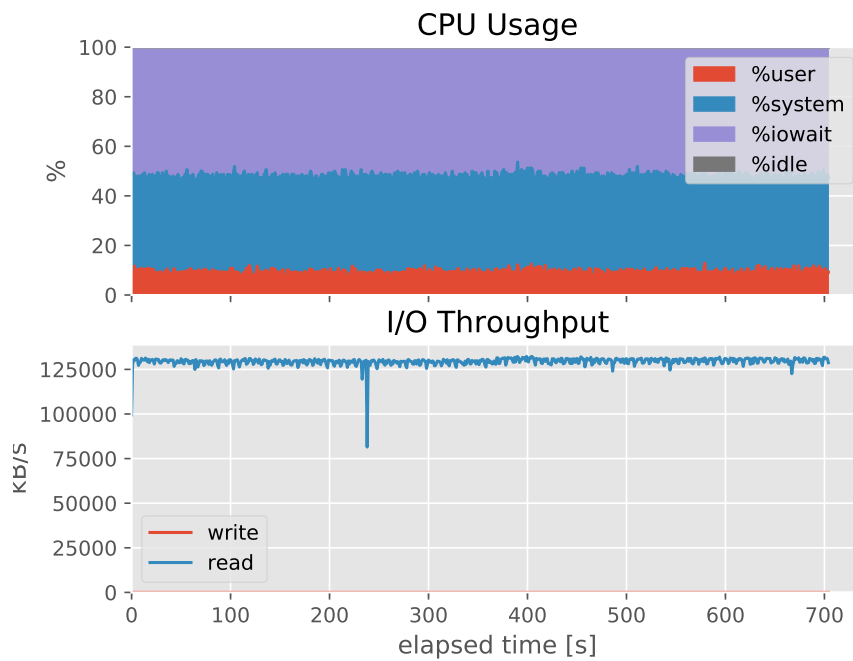


図 61: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット ( $\text{Rand} \cdot \sigma = 4e-3$ )

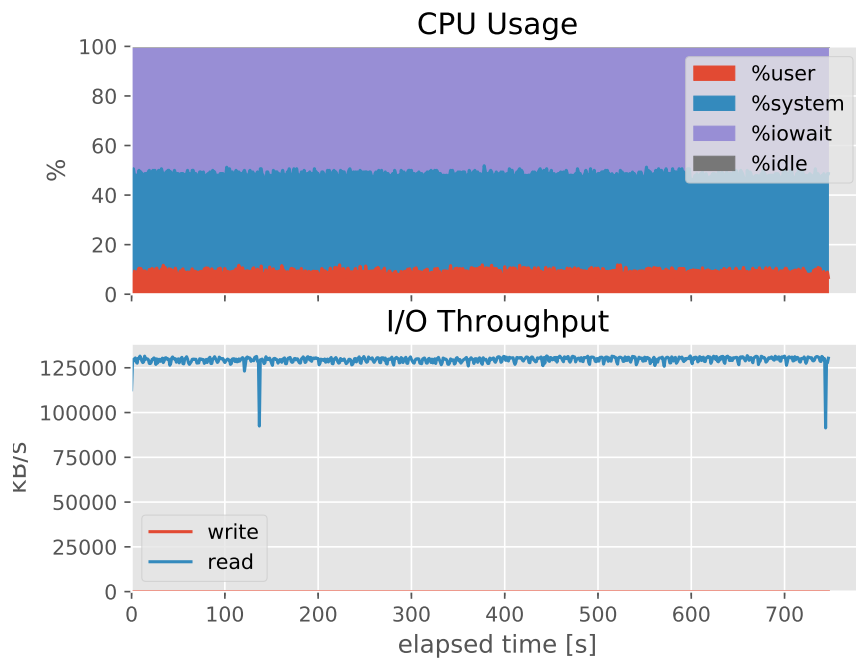


図 62: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット ( $\text{Rand} \cdot \sigma = 5e-3$ )

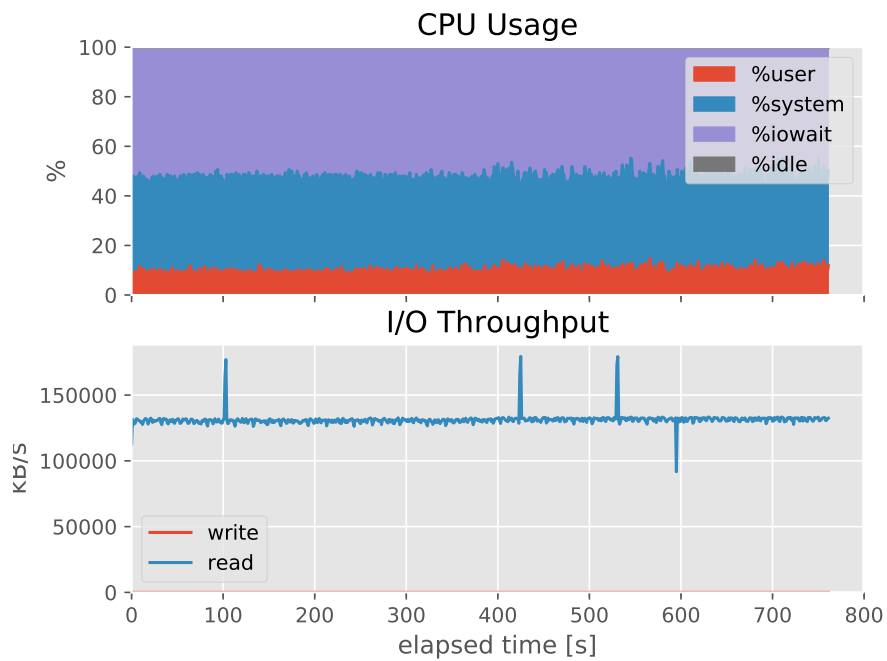


図 63: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット ( $\text{Rand} \cdot \sigma = 6e-3$ )

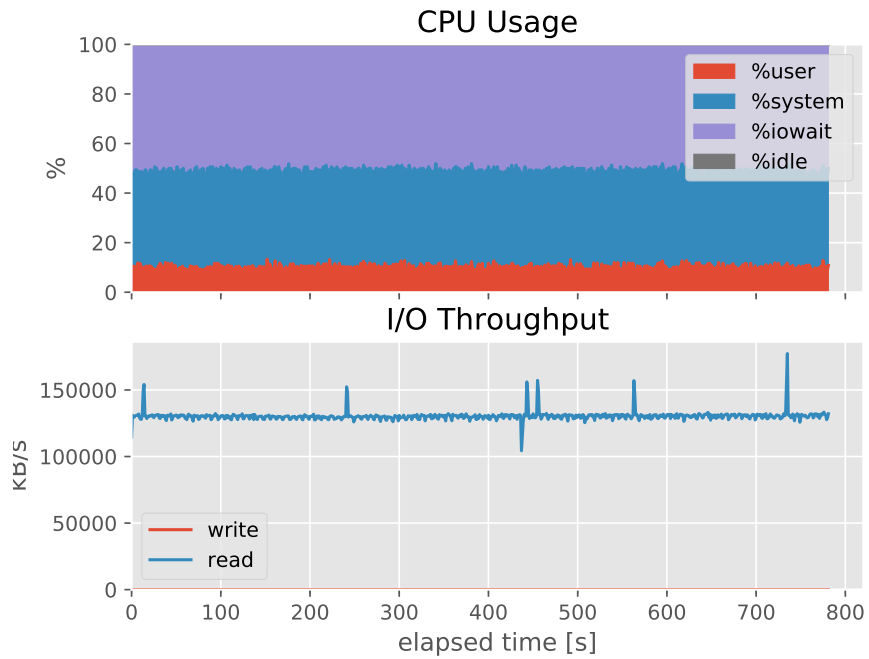


図 64: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット ( $\text{Rand} \cdot \sigma = 7e-3$ )

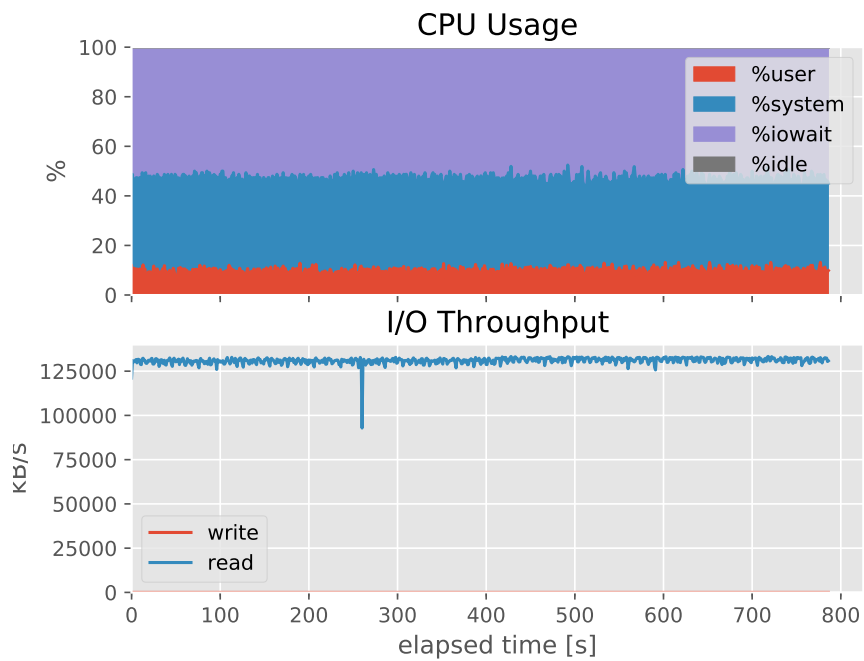


図 65: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット ( $\text{Rand} \cdot \sigma = 8e-3$ )

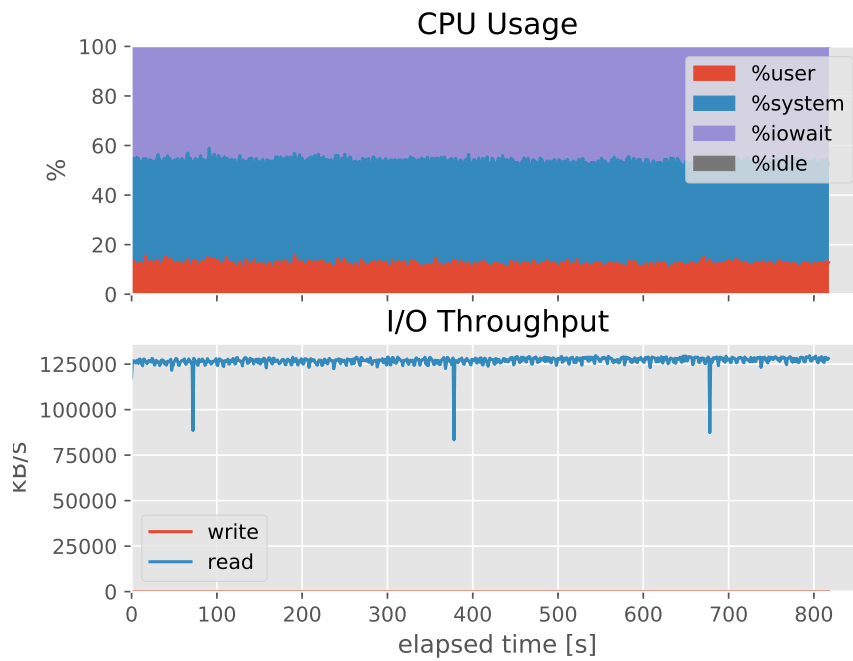


図 66: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット ( $\text{Rand} \cdot \sigma = 9e-3$ )

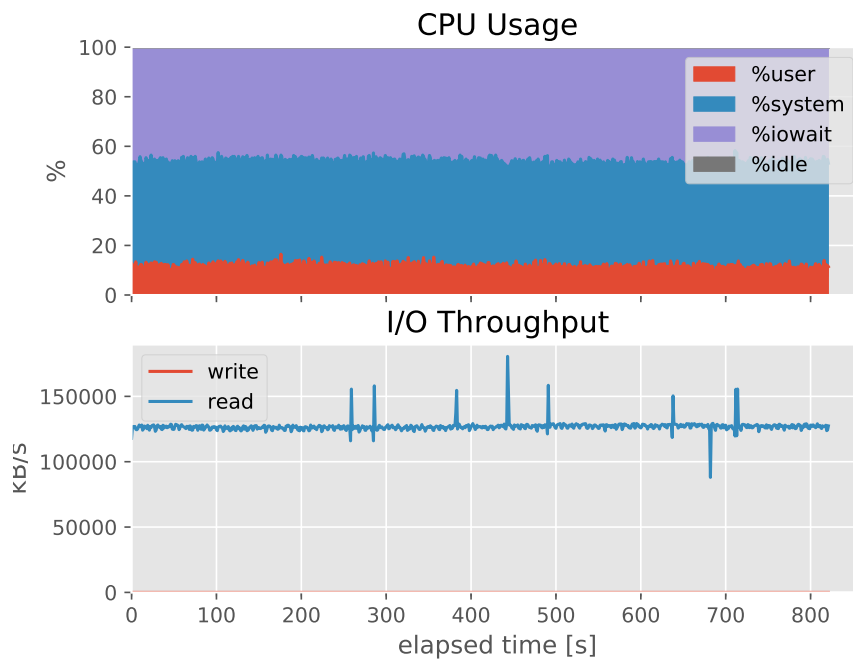


図 67: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット ( $\text{Rand} \cdot \sigma = 0.01$ )



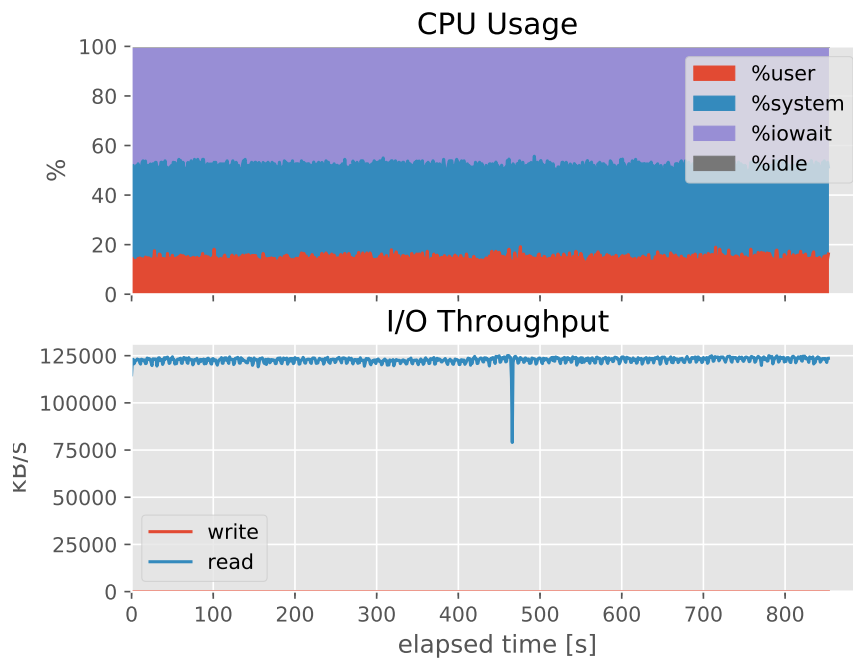


図 68: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット ( $\text{Rand} \cdot \sigma = 0.02$ )

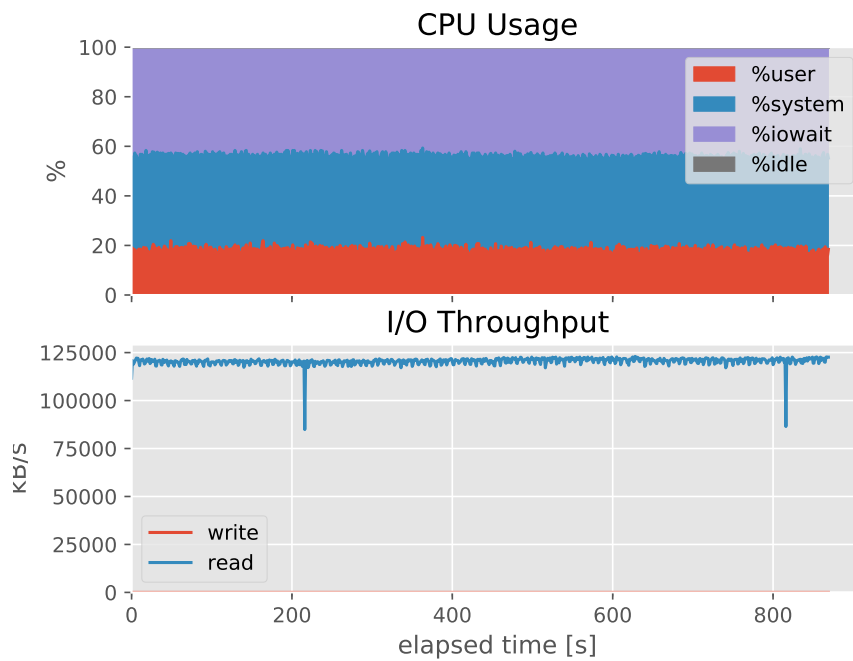


図 69: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット ( $\text{Rand} \cdot \sigma = 0.03$ )

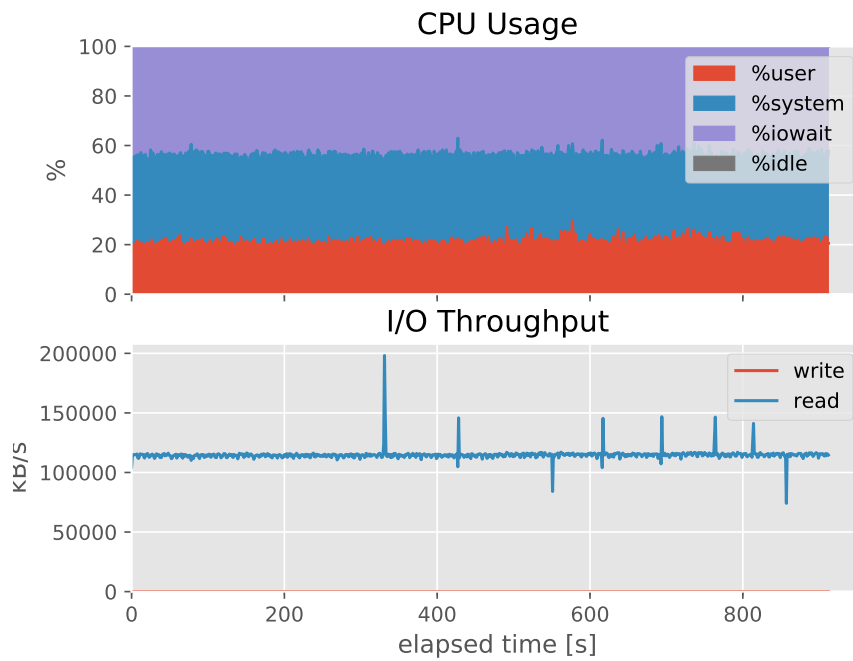


図 70: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット ( $\text{Rand} \cdot \sigma = 0.04$ )

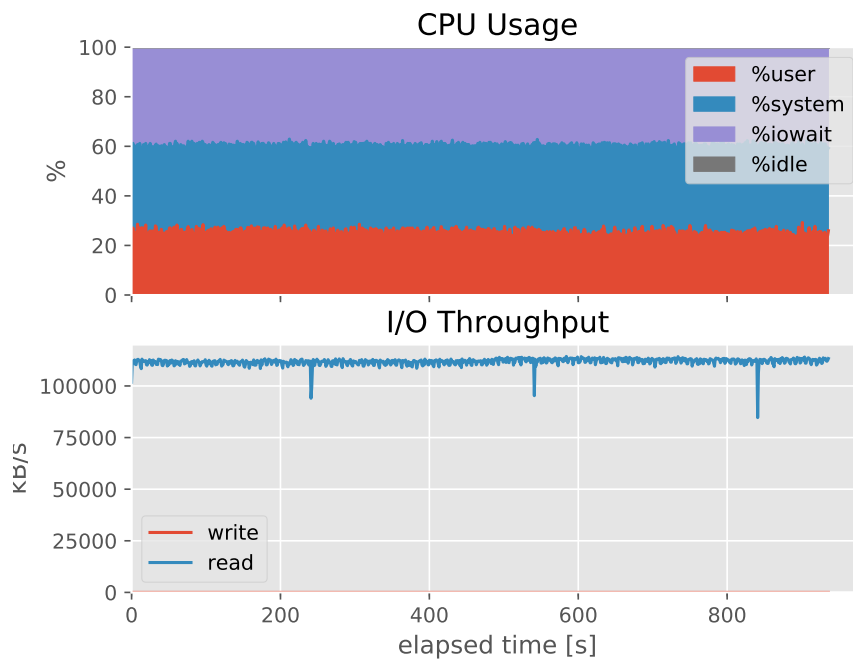


図 71: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット ( $\text{Rand} \cdot \sigma = 0.05$ )

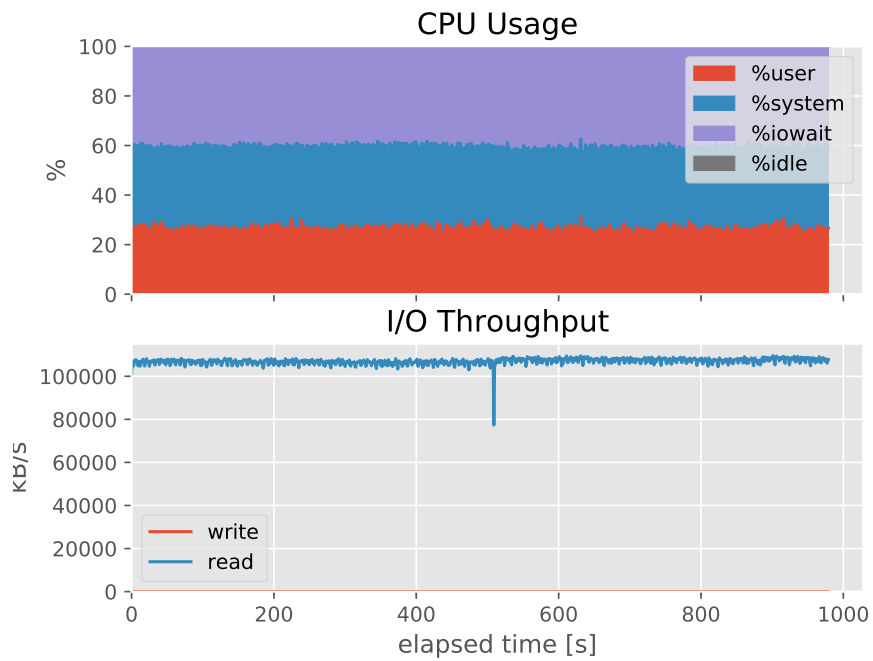


図 72: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット ( $\text{Rand} \cdot \sigma = 0.06$ )

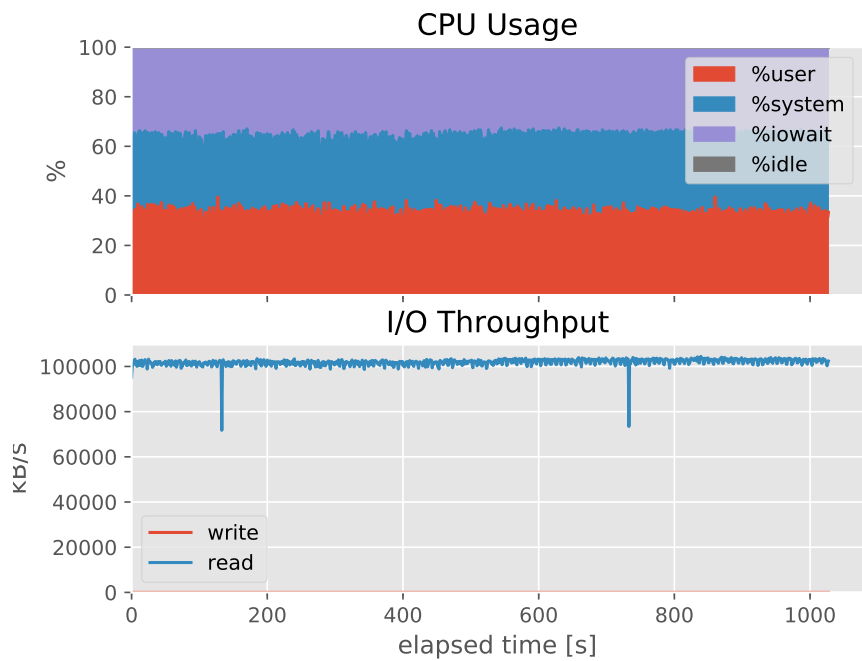


図 73: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット ( $\text{Rand} \cdot \sigma = 0.07$ )

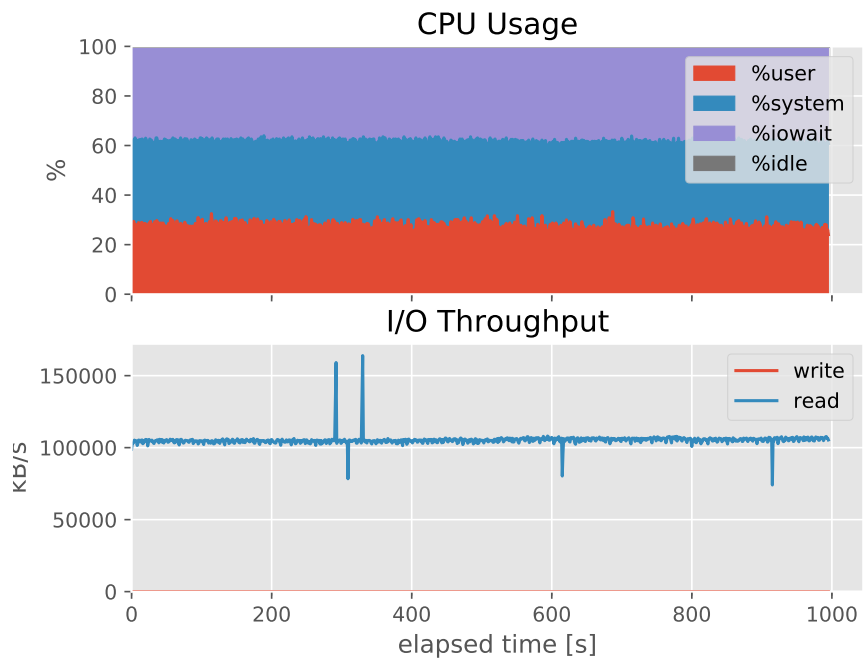


図 74: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット ( $\text{Rand} \cdot \sigma = 0.08$ )

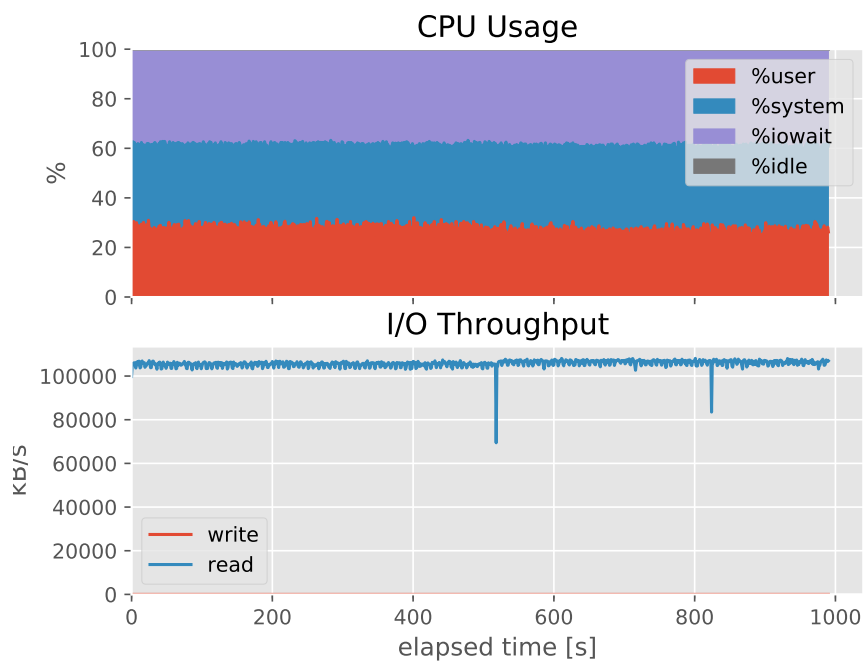


図 75: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット ( $\text{Rand} \cdot \sigma = 0.09$ )

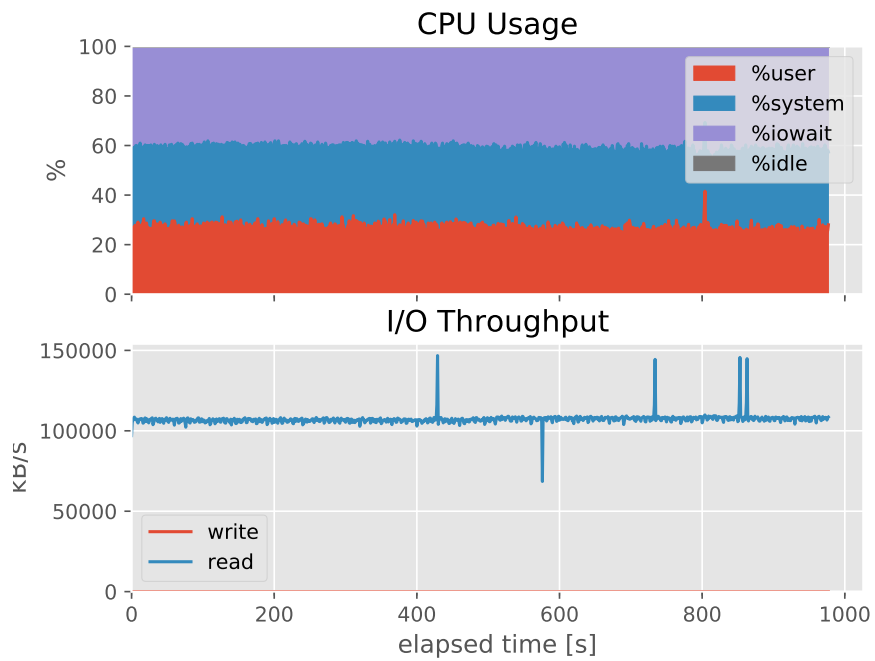


図 76: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand ·  $\sigma = 0.1$ )

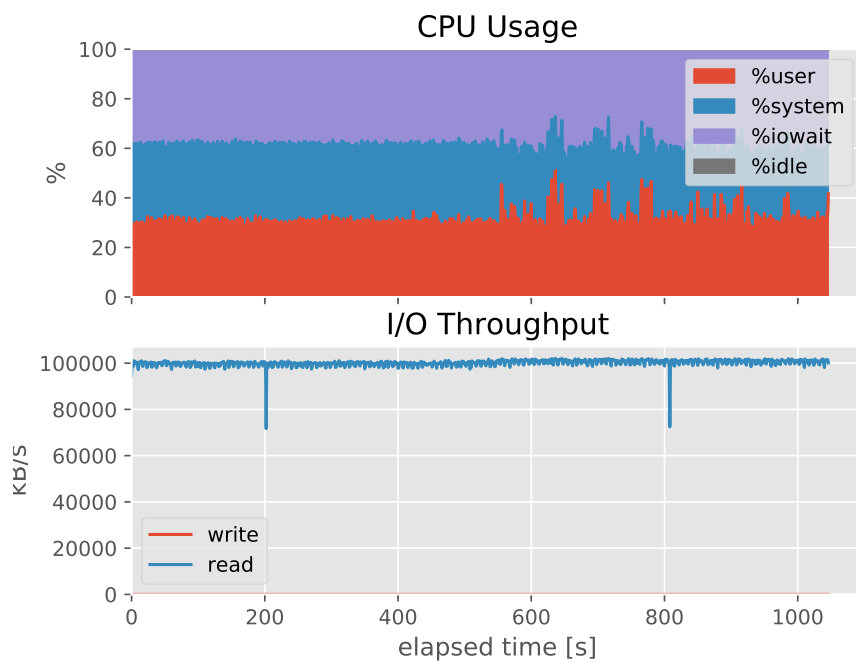


図 77: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand ·  $\sigma = 0.2$ )

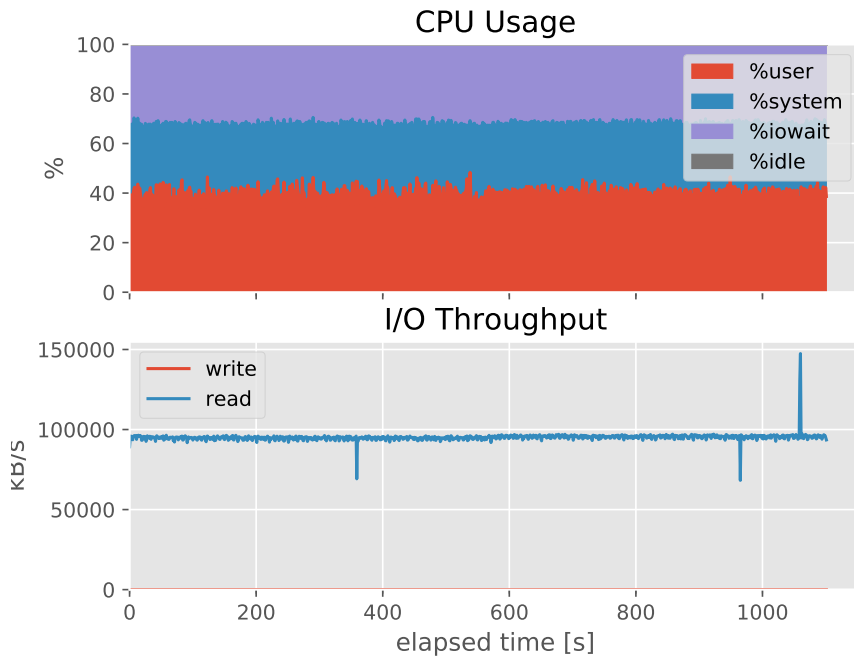


図 78: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand ·  $\sigma = 0.3$ )

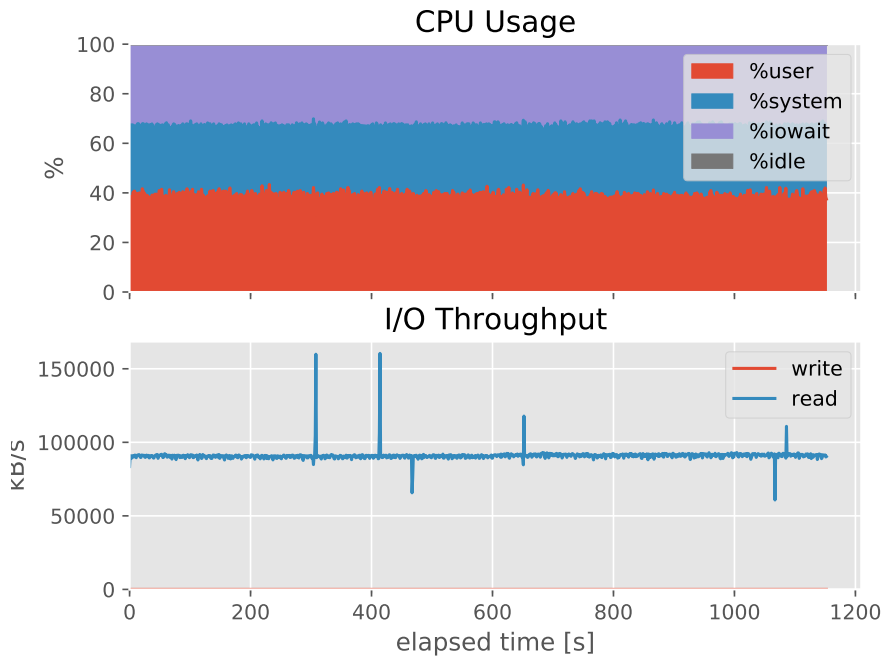


図 79: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand ·  $\sigma = 0.4$ )

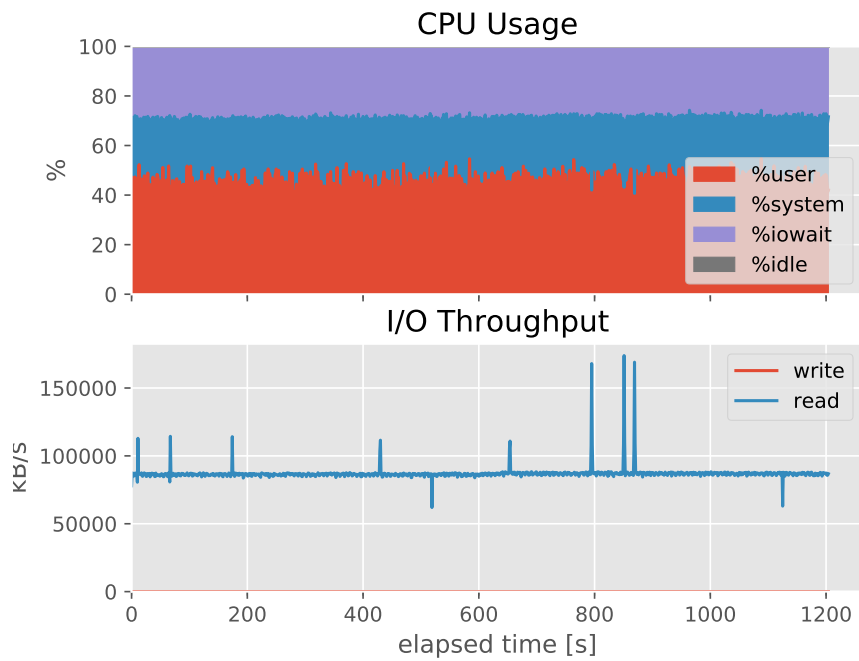


図 80: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand ·  $\sigma = 0.5$ )

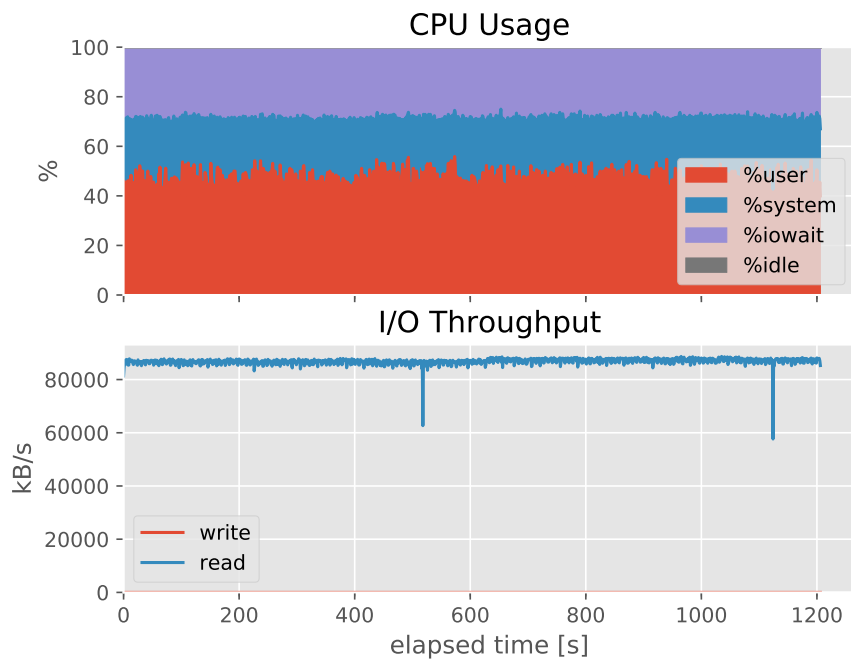


図 81: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand ·  $\sigma = 0.6$ )

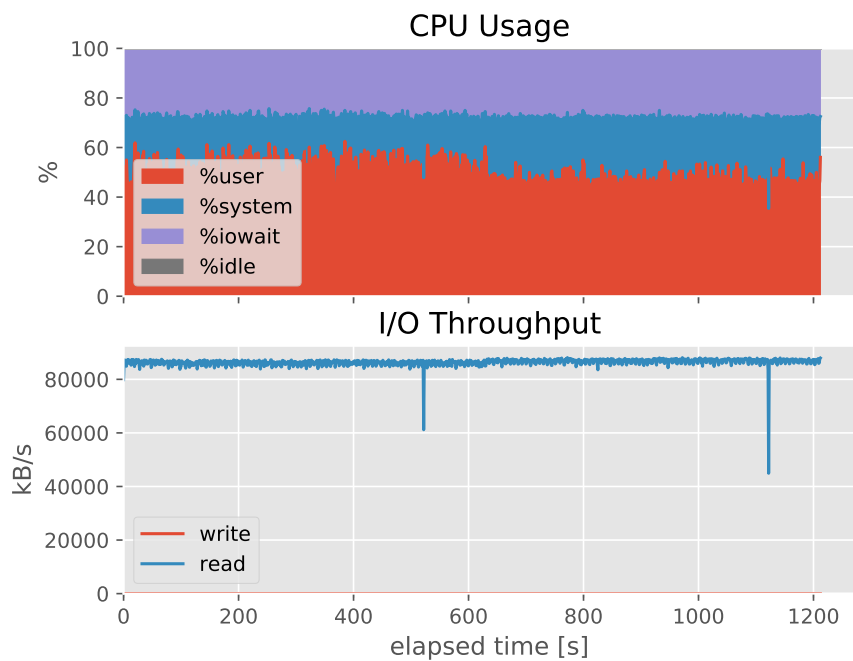


図 82: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand ·  $\sigma = 0.7$ )

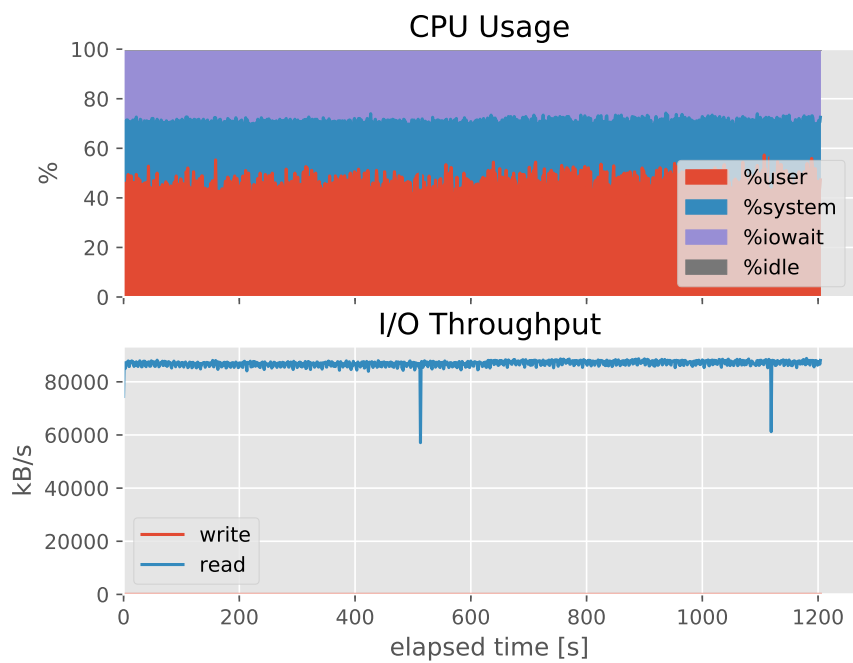


図 83: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand ·  $\sigma = 0.8$ )



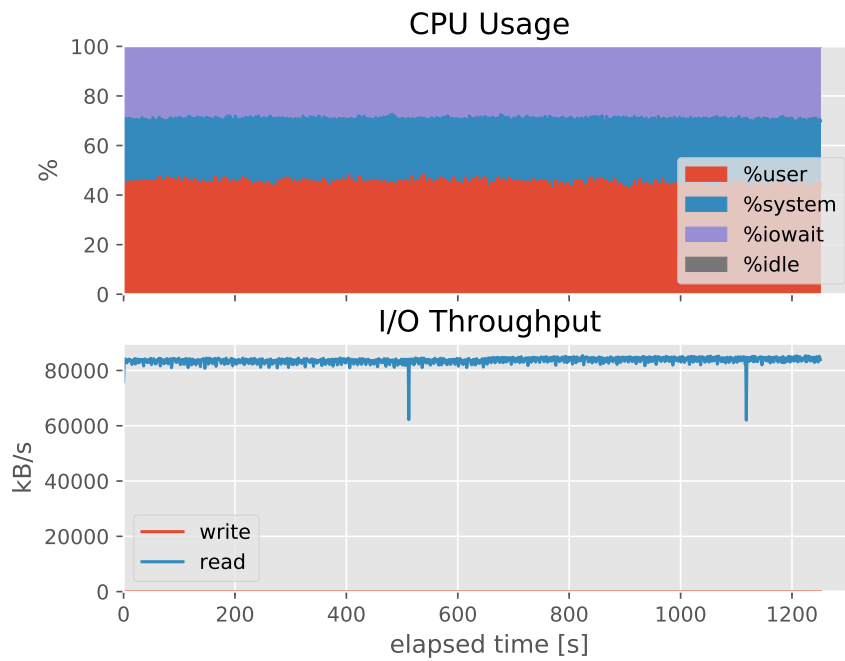


図 84: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand ·  $\sigma = 0.9$ )

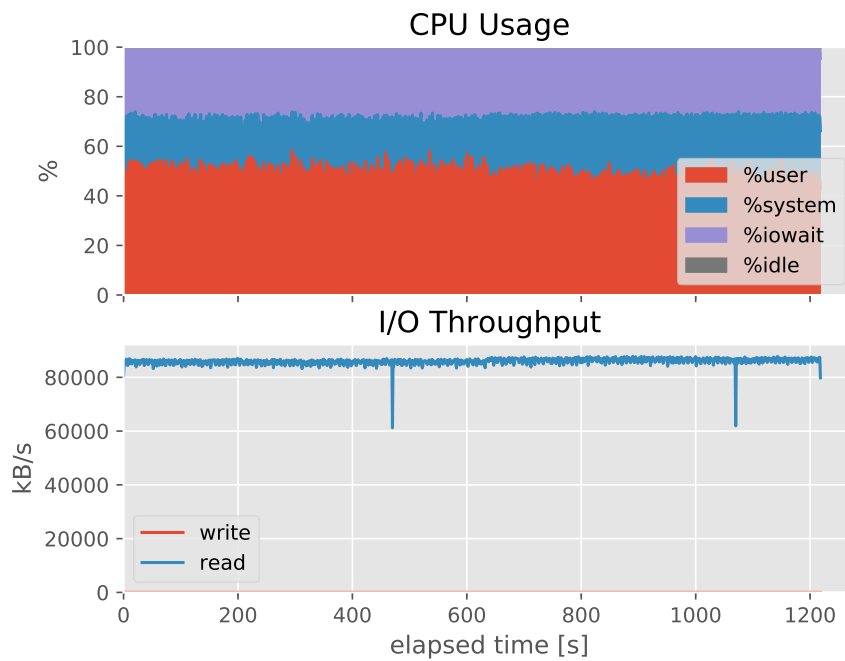


図 85: 単純問合せ実行時の毎秒 CPU 利用率・I/O スループット (Rand ·  $\sigma = 1$ )

## 5.5 データベース管理システムの再起動時の問合せ実行待ち時間

次にデータベース管理システムの再起動過程を模擬し、再起動中に問合せを受け付けた状況を想定して、その問合せ処理の実行待ち時間を計測した。問合せ実行待ち時間の結果を図 86 と図 87 に示す。

図において、横軸は再起動からの経過時間を表し、ログ適用を実行した時間を青色の棒グラフ、問合せ処理を実行した時間をオレンジ色の棒グラフとして表している。上 2 つが従来技術でログ適用を待って問合せ処理を実行した場合で、ログ適用の青色の棒の右に問合せ処理のオレンジ色の棒が位置していて、ログ適用完了後に問合せ処理を実行している様子を表している。中央 2 つは早期ログ適用技法を用いてログ適用と並行して問合せ処理を実行した場合で、再起動と同時に問合せ処理を受け付けた状況である。青色の棒とオレンジ色の棒が上下に重なっていることが並行に動作している様子を表していて、問合せ処理の受付直後から問合せ処理の実行を開始している。このとき問合せ処理はログ適用よりも早く完了するが、その応答は早期ログ適用技法によってログ適用後の一貫性を回復した状態に対する問合せ応答と等しい。下 2 つは中央 2 つと同様に早期ログ適用技法を用いた場合だが、再起動の 50 秒に問合せ処理を受け付けた状況である。同様に、問合せ処理の受付直後に問合せ処理の実行を開始していることが見て取れる。

テストログはいずれも共通で、生成パターンとしてランダムアクセスを用いた。テスト問合せとしてランダムアクセスで選択率  $1e-5$  と選択率  $1e-6$  を用いており、図 86 は選択率  $1e-5$ 、図 87 は選択率  $1e-6$  の場合の結果になっている。

選択率  $1e-5$  のテスト問合せの場合で、問合せ実行待ち時間は、ログ適用手法を整列ログ適用にすることで 19% の短縮となり、早期ログ適用技法も用いることで、57% の短縮となることが確認できた。また早期ログ適用技法を用いることによるログ適用自体のオーバーヘッドは、逐次ログ適用の場合で 2%、整列ログ適用の場合で 11% となった。

再起動過程におけるログ適用と問合せ処理実行時の毎秒の CPU 利用率と I/O スループットを図 88 から図 99 に示す。図 88 から図 93 までが選択率  $1e-5$  のテスト問合せの場合で、図 94 から図 99 までが選択率  $1e-6$  のテスト問合せの場合である。従来の再起動過程、早期ログ適用技法を用いた再起動過程で再起動と同時に問合せ受付、早期ログ適用技法を用いた再起動過程で再起動の 50 秒後に問合せ受付の状況について、逐次ログ適用と整列ログ適用を行った場合を順に表示している。

従来の再起動過程の I/O スループットの変動からは、読み込みと書き込みの I/O

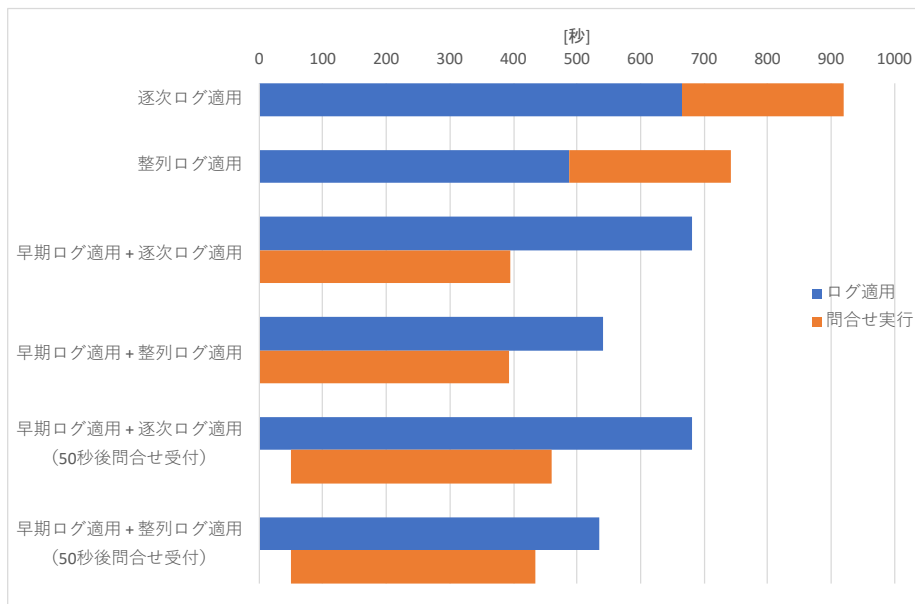


図 86: 再起動時の問合せ実行待ち時間 (ログ: Rand) (問合せ:  $\text{Rand} \cdot \sigma = 1e-5$ )

が同数発行されるログ適用の後で、読み込み I/O のみが発行される問合せ処理が実行されている様子が見て取れる。早期ログ適用技法を用いた再起動過程の I/O スループットの変動からは、問合せの受付と同時に読み込み (read) の I/O スループットが書き込み (write) に対して増加していて、ログ適用と並行して問合せ処理を実行している様子が見て取れる。

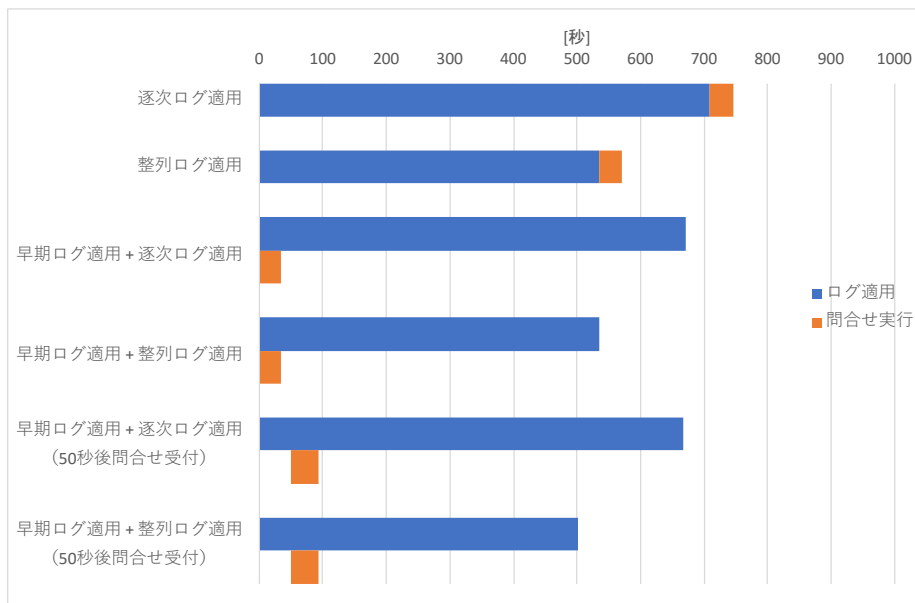


図 87: 再起動時の問合せ実行待ち時間 (ログ: Rand) (問合せ:  $\text{Rand} \cdot \sigma = 1e-6$ )

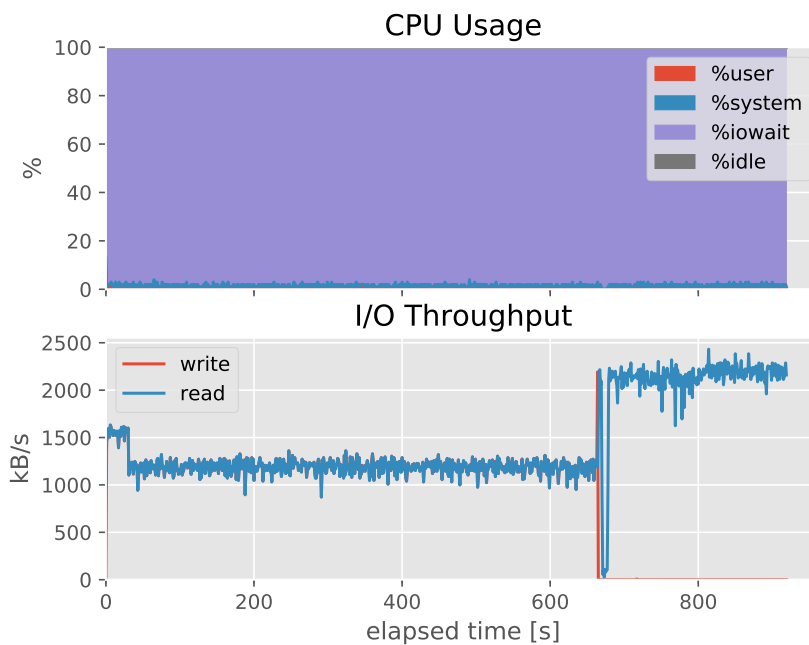


図 88: 従来の再起動過程の毎秒 CPU 利用率・I/O スループット (ログ: Rand・逐次ログ適用) (問合せ:  $\text{Rand} \cdot \sigma = 1e-5$ )

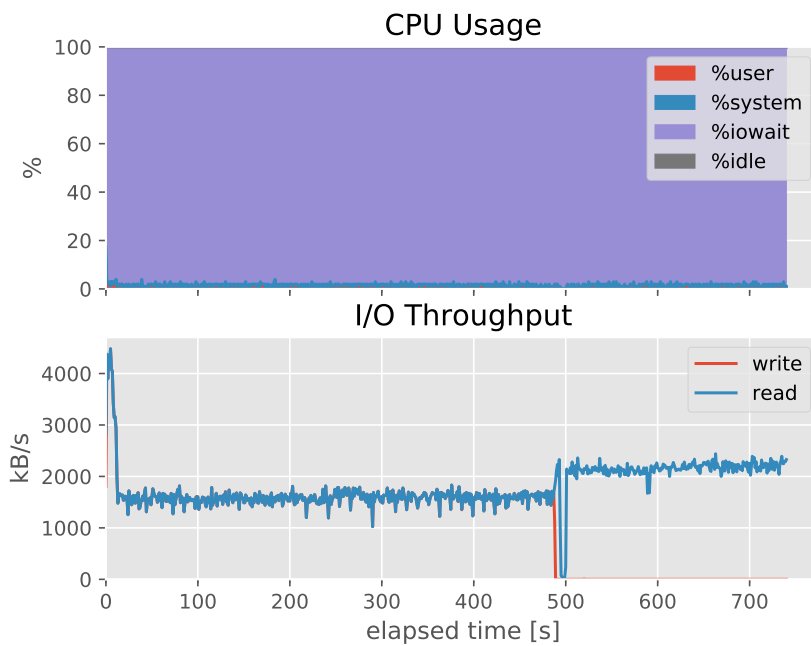


図 89: 従来の再起動過程の毎秒 CPU 利用率・I/O スループット (ログ: Rand・整列ログ適用) (問合せ:  $\text{Rand} \cdot \sigma = 1e-5$ )

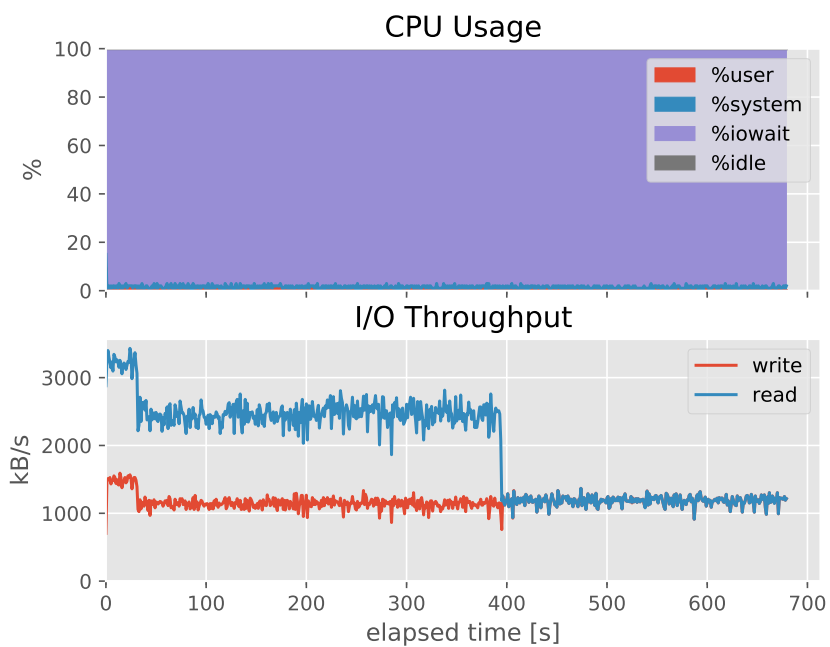


図 90: 早期ログ適用を用いた再起動過程の毎秒 CPU 利用率・I/O スループット (ログ: Rand・逐次ログ適用) (問合せ:  $\text{Rand} \cdot \sigma = 1e-5$ , 同時受付)

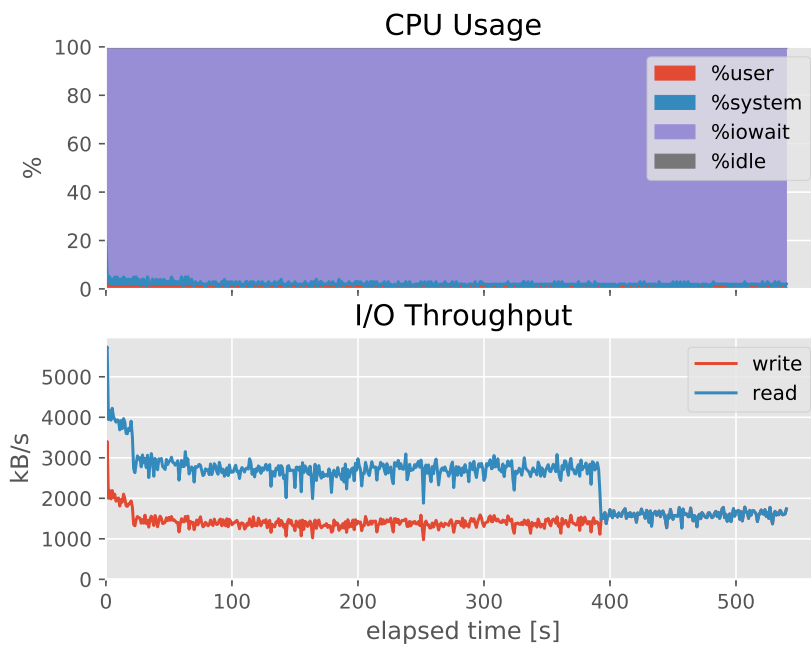


図 91: 早期ログ適用技法を用いた再起動過程の毎秒 CPU 利用率・I/O スループット (ログ: Rand・整列ログ適用) (問合せ:  $\text{Rand} \cdot \sigma = 1e-5$ , 同時受付)

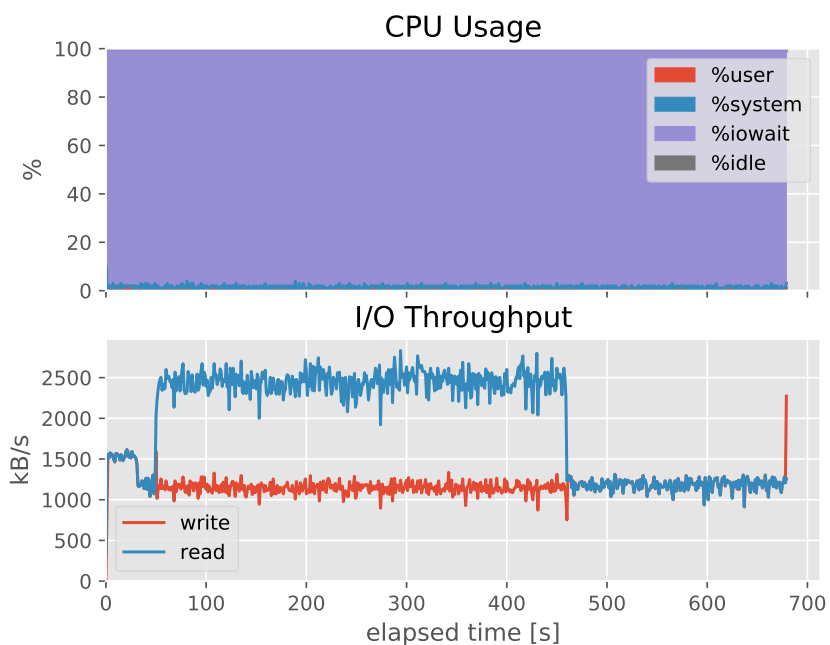


図 92: 早期ログ適用技法を用いた再起動過程の毎秒 CPU 利用率・I/O スループット (ログ: Rand・逐次ログ適用) (問合せ:  $\text{Rand} \cdot \sigma = 1e-5$ , 50 秒後受付)

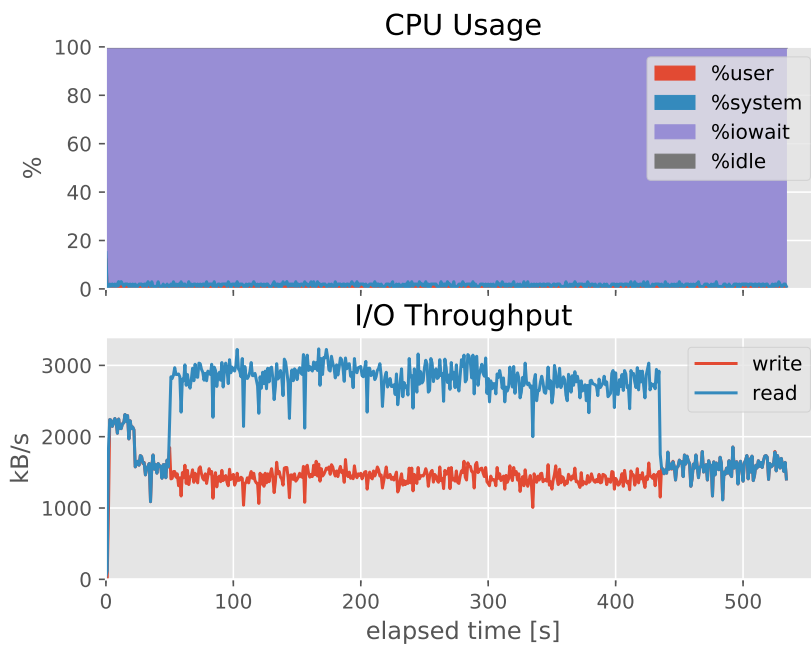


図 93: 早期ログ適用技法を用いた再起動過程の毎秒 CPU 利用率・I/O スループット (ログ: Rand・整列ログ適用) (問合せ:  $\text{Rand} \cdot \sigma = 1e-5$ , 50 秒後受付)

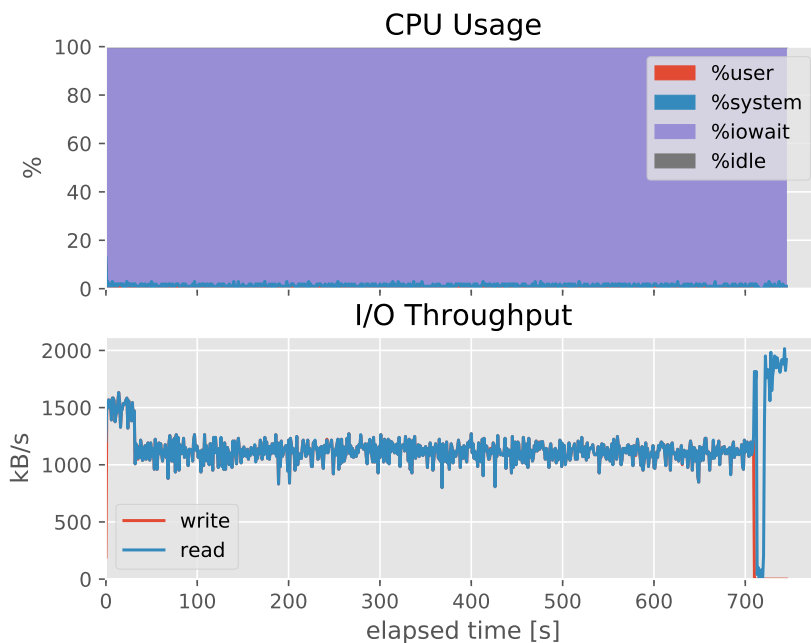


図 94: 従来の再起動過程の毎秒 CPU 利用率・I/O スループット (ログ: Rand・逐次ログ適用) (問合せ:  $\text{Rand} \cdot \sigma = 1e-6$ )

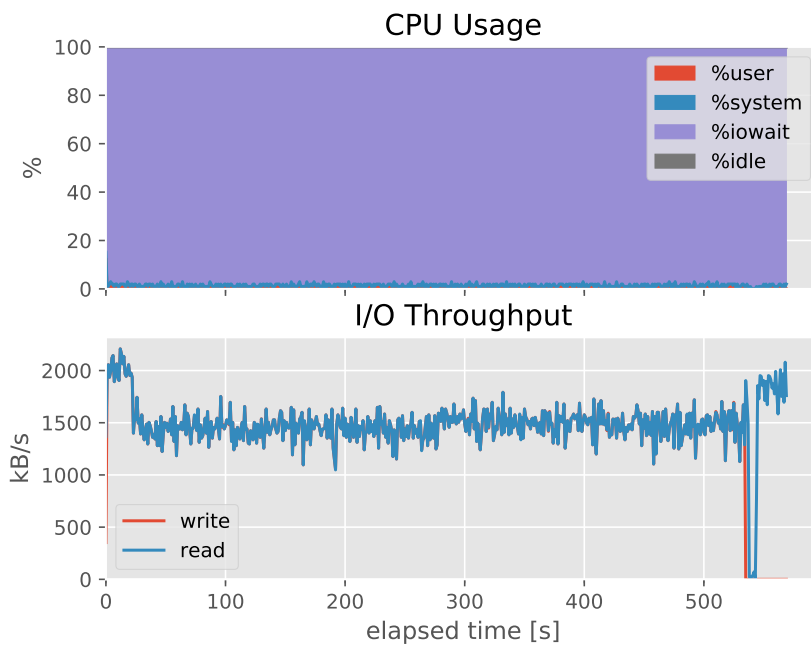


図 95: 従来の再起動過程の毎秒 CPU 利用率・I/O スループット (ログ: Rand・整列ログ適用) (問合せ:  $\text{Rand} \cdot \sigma = 1e-6$ )

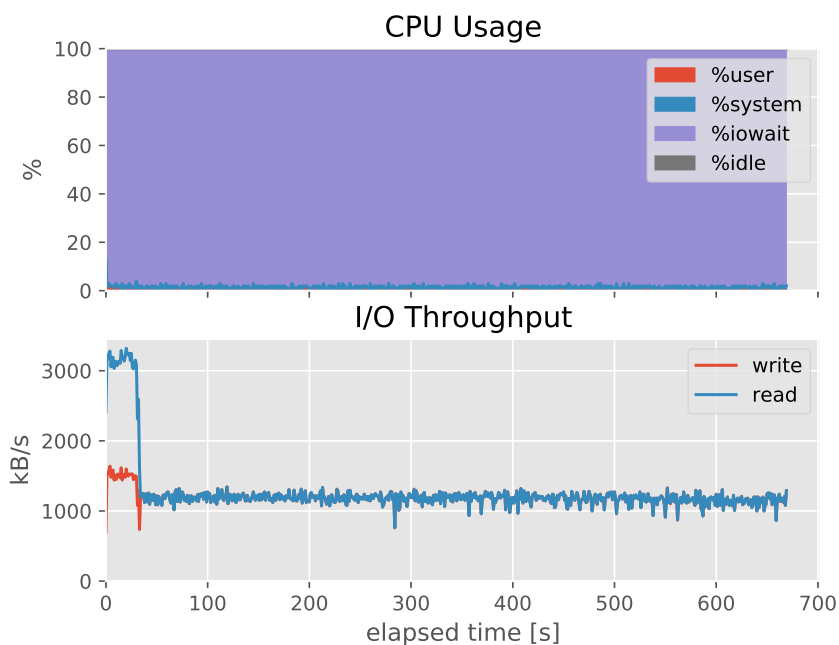


図 96: 早期ログ適用技法を用いた再起動過程の毎秒 CPU 利用率・I/O スループット (ログ: Rand・逐次ログ適用) (問合せ:  $\text{Rand} \cdot \sigma = 1e-6$ , 同時受付)



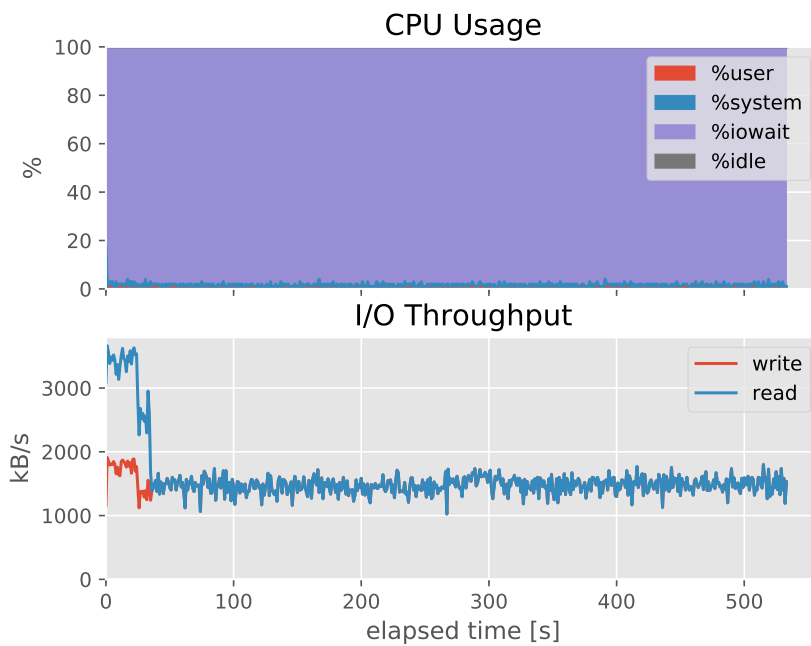


図 97: 早期ログ適用技法を用いた再起動過程の毎秒 CPU 利用率・I/O スループット (ログ: Rand・整列ログ適用) (問合せ:  $\text{Rand} \cdot \sigma = 1e-6$ , 同時受付)

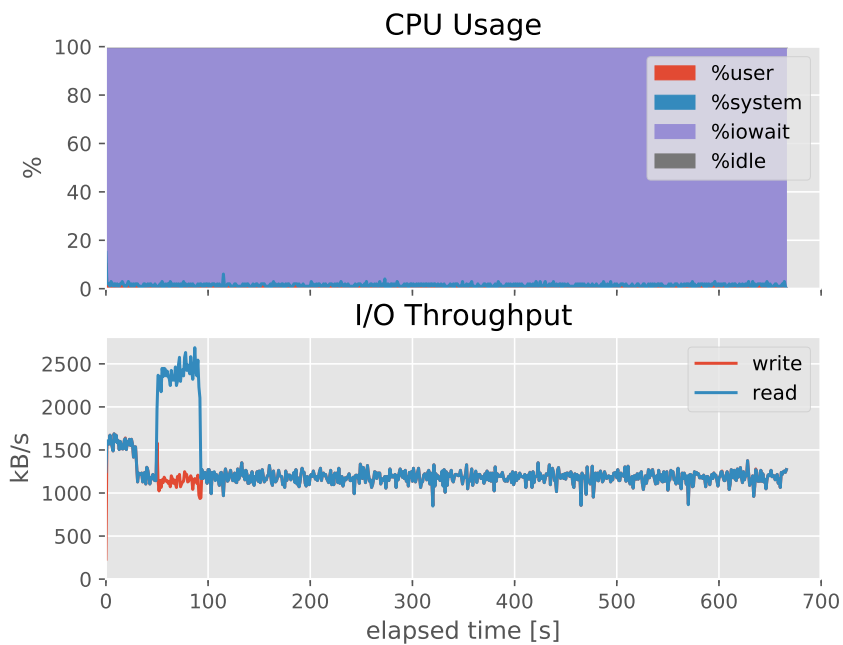


図 98: 早期ログ適用技法を用いた再起動過程の毎秒 CPU 利用率・I/O スループット (ログ: Rand・逐次ログ適用) (問合せ:  $\text{Rand} \cdot \sigma = 1e-6$ , 50 秒後受付)

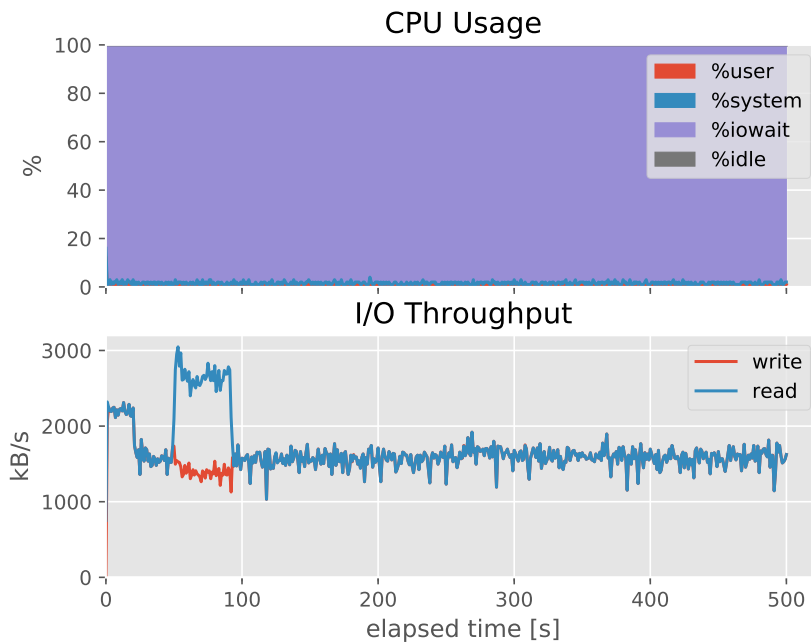


図 99: 早期ログ適用技法を用いた再起動過程の毎秒 CPU 利用率・I/O スループット (ログ: Rand・整列ログ適用) (問合せ: Rand・ $\sigma = 1e-6$ , 50 秒後受付)

実験結果をまとめると、まず、整列ログ適用は逐次ログ適用よりもデータベース管理システム再起動時の問合せ実行待ち時間を 19%短縮した。早期ログ適用技法を用いた時、再起動と同時に問合せ実行開始した場合で、問合せ実行待ち時間を 57%短縮した。早期ログ適用技法による、ログ適用の実行時間に与えるオーバーヘッドは高々 11%だった。

早期ログ適用による問合せ実行待ち時間の短縮効果は、単なる整列ログ適用による短縮効果を上回り、両者の組合せによりさらなる短縮効果が得られた。以上のように、早期ログ適用技法によって他では得られない問合せ実行待ち時間の短縮効果が得られることが確認できた。

## 6 まとめと今後の課題

本論文は、早期ログ適用によってログ適用中に発行された問合せ処理の実行待ち時間が改善する効果を、実験により明らかにした。

一般にデータベース管理システムは、ライトアヘッドロギングとチェックポイントを採用しており、障害等により再起動されると、データベースを格納するボリュームならびにログのディレクトリ情報等を探索し、最新のチェックポイントを同定し、当該チェックポイント以降に記録されたログをデータベースに適用し、データベースを回復し、一貫性のある状態となってから問合せ処理を受け付け、これの処理を実行する。よって、データベース管理システムに障害等が生じ再起動される場合、データベースを回復するまでの間、データベース管理システムは新たな問合せを受け付けて処理することができない。これはデータベース管理システムの可用性を低下させる1つの要因となっている。

早期ログ適用は、データベース管理システムが障害等により再起動された際に、ログの適用によるデータベースの回復を待つことなく、先行的に問合せ処理を実行可能とする技法であり、問合せがデータベースからページを取得する際には、当該ページが既に回復済みであって既にコミットされたトランザクションによる全ての更新が適用されているかどうかを判定し、そうであれば当該ページをそのまま利用し、そうでない場合には当該ページに未適用のログを取得することによりコミットされたトランザクションによる全ての更新が適用されている状態に回復させてから当該ページを利用する。

著者は、早期ログ適用機能を備えたデータベースエンジンを試作し、複数の更新偏りパターンの更新トランザクションによって生成された一連のログを対象として、当該ログの適用中に複数のデータアクセス偏りパターンを有する問合せを発行し、それぞれの組合せで、ログ適用ならびに問合せ処理に要する実行時間や実行中に各種資源利用量を計測し、問合せ処理の実行待ち時間の変化を評価した。この結果、早期ログ適用技法を用いることにより、ログ適用中に発行された問合せ処理の実行待ち時間が改善することを確認した。

上記の実験では比較的簡単なデータセットを用いて評価を行ったが、今後はTPC-C[5]をはじめとする標準的なベンチマークや現実のデータセットならびに負荷を用いてその有効性を確認する必要がある。また、オープンソースのデータベース管理システム等へ実装することにより、実装上の工夫や周辺ソフトウェアとの親

和性等を丁寧に検証していく必要がある。

# 謝辞

非常に多くの方々から多大なご指導，ご協力，励ましを頂き，本論文を完成させることができました。

はじめに，指導教官の喜連川優教授に深く感謝いたします。

本研究を進めるにあたって，私が師事する合田和生特任准教授には様々なご指導をしていただくとともに，研究のみならず様々な相談に乗っていただきました。この場を借りて感謝の意を表したいと思います。

豊田正史准教授を始め研究室のスタッフの皆様には，本研究に関する議論や論文の執筆指導など，研究生活に関わる様々な面で丁寧かつ熱心にご指導していただきましたことに感謝いたします。

## 参考文献

- [1] Ramez Elmasri and Shamkant Navathe. *Fundamentals of database systems*. Addison-Wesley Publishing Company, 2010.
- [2] Jim Gray and Andreas Reuter. *Transaction processing: concepts and techniques*. Elsevier, 1992.
- [3] C Mohan, Don Haderle, Bruce Lindsay, Hamid Pirahesh, and Peter Schwarz. Aries: a transaction recovery method supporting fine-granularity locking and partial rollbacks using write-ahead logging. *ACM Transactions on Database Systems (TODS)*, Vol. 17, No. 1, pp. 94–162, 1992.
- [4] 塚井知之, 加藤比呂武. Oracle の高可用性技術とミッションクリティカル・システムへの適用. 電子情報通信学会技術研究報告. DE, データ工学, Vol. 107, No. 254, pp. 63–68, 2007.
- [5] Transaction Processing Council. <http://www.tpc.org/>.
- [6] Dennis Shasha and Philippe Bonnet. *Database tuning: principles, experiments, and troubleshooting techniques*. Elsevier, 2002.
- [7] Anant Jhingran and Pratap Khedkar. Analysis of recovery in a database system using a write-ahead log protocol. In *Proceedings of the 1992 ACM SIGMOD International Conference on Management of Data, San Diego, California, USA, June 2-5, 1992.*, pp. 175–184, 1992.
- [8] Henry F Korth, Eliezer Levy, and Abraham Silberschatz. *A formal approach to recovery by compensating transactions*. University of Texas at Austin, Department of Computer Sciences, 1990.
- [9] Michael J. Franklin, Michael J. Zwillig, C. K. Tan, Michael J. Carey, and David J. DeWitt. Crash recovery in client-server EXODUS. In *Proceedings of the 1992 ACM SIGMOD International Conference on Management of Data, San Diego, California, USA, June 2-5, 1992.*, pp. 165–174, 1992.

- [10] Vijay Kumar and Meichun Hsu. *Recovery mechanisms in database systems*. Prentice Hall PTR, 1997.
- [11] Hong-Tai Chou and David J DeWitt. An evaluation of buffer management strategies for relational database systems. *Algorithmica*, Vol. 1, No. 1-4, pp. 311–336, 1986.
- [12] Joaquín Pérez, et al. Least likely to use: a new page replacement strategy for improving database management system response time. In *International Computer Science Symposium in Russia*, pp. 514–523. Springer, 2006.
- [13] Makoto Yui, Jun Miyazaki, Shunsuke Uemura, and Hayato Yamana. Nb-gclock: A non-blocking buffer management based on the generalized clock. In *Data Engineering (ICDE), 2010 IEEE 26th International Conference on*, pp. 745–756. IEEE, 2010.
- [14] Jeffrey Scott Vitter. Faster methods for random sampling. *Communications of the ACM*, Vol. 27, No. 7, pp. 703–718, 1984.

# 発表文献

1. 谷川祐一, 合田和生, 喜連川優. 早期ログ適用技法が再起動時の問合せ実行待ちに与える影響の実験的考察. 第11回 Web とデータベースに関するフォーラム (WebDB Forum 2018) 電子情報通信学会データ工学研究会セッション (2018.09) (発表予定)