PITSTOP: Pausable Data Inspection for Graph Databases (Supplementary Material)

C	$::= \langle D; O; R\rangle$	*configuration*
O	$::= \overrightarrow{\ell \mapsto o}$	*operations*
o	$::= \texttt{add}\ v \mid \texttt{update}\ f\ k \mid \texttt{query}\ f\ k$	*operation*
D	$::= \overrightarrow{L}$	*data*
L	$::= \langle N; O\rangle$	*legged data*
R	$::= \overrightarrow{\ell \mapsto v}$	*result store*
N	$::= \mathbf{N}\langle k; v\rangle$	*node value*

Figure 1: Runtime Definitions

1. Syntax and Runtime Structures

1.1. Syntax

Notations We use $\overrightarrow{\sigma}^m$ to represent $[\sigma_1, \ldots, \sigma_m]$ and $[]$ as an empty sequence. When m does not matter, we also write $\overrightarrow{\sigma}^m$ as $\overrightarrow{\sigma}$. When a sequence μ takes the form of $\overrightarrow{\sigma \mapsto \sigma'}^m$, we define $\mu(\sigma_i)$ as σ'_i for some $1 \leq i \leq m$; $dom(\mu)$ as $\overline{\sigma}^m$; and $ran(\mu)$ as $\overline{\sigma'}^m$. We use $\mu_1 \uplus \mu_2$ to represent the standard sequence concatenation of μ_1 and μ_2, defined iff $dom(\mu_1) \cap dom(\mu_2) = \{\}$. Binary operator $\sigma :: \Sigma$ prepends σ to sequence Σ as the head, and binary operator $\Sigma +\!\!+ \Sigma'$ concatenates Σ and Σ' together. We elide their definitions here.

For operations, we introduce a convenience function $\odot$ that computes the keys of nodes where the operation is intended for realization:

Definition 1 (Operation Target)**.** *The function $\odot(o)$ computes the* target *of the operation o, defined as k if $o = \texttt{map}\ f\ k$ or $o = \texttt{fold}\ f\ k$. The operator is undefined for* add.

Definition 2 (Operation Stream Addition and Result Store Addition)**.** *The (overloaded) $\blacktriangleleft$ operator appends a stream unit to the configuration, or appends a stream unit to a non-empty backend, or adds results to the configuration:*

$$\langle D; O; R\rangle \blacktriangleleft U \triangleq \langle D; O +\!\!+ [U]; R\rangle$$
$$\langle N; O\rangle :: D \,\underline{\blacktriangleleft}\, U \triangleq \langle N; O +\!\!+ [U]\rangle :: D$$
$$\langle D; O; R\rangle \lhd R' \triangleq \langle D; O; R +\!\!+ R'\rangle$$

The definition above says that any addition to an operation stream — be it a top-level operation stream or a streamlet — must be *appended*. As we shall see in the operational semantics, any *removal* from the operation stream will be from the head. It is through this consistent access pattern that the chronological order of the operations is preserved in our semantics.

2. DON Calculus Operational Semantics

The reduction relation $C \to C'$ in Fig. 3 says that configuration C one-step reduces to configuration C'. We use $\to^*$ to represent the reflexive and transitive closure of $\to$. Evaluation contexts are defined in Fig. 2.

3. Meta-Theory

We define equivalence relation $C \sim C'$ by the rules in Fig. 4, where helper relation $D \overset{b}{\sim} D'$ says that the payload and edge list of all stations in the backends are term equivalent, and that the intention of the delayed operations in the stations are equivalent. The most interesting parts of the $\overset{b}{\sim}$ relation is the *write effect* and *read result* equivalence check. These are defined with helper functions *writes* and *reads*, respectively.

For the rest of this section, we only consider finite reduction sequences. There are cases where a reduction sequence involving $\to$ can be infinite, but we are not concerned with proving the confluence of divergent reductions.

We use the following notation: $C_x \overset{\sim}{\downarrow} C_y \Leftrightarrow \exists C_u, C_v. C_x \to^* C_u \wedge C_y \to^* C_v \wedge C_u \sim C_v$.

Lemma 1 (Local Confluence Modulo $\sim$ (P1))**.** *For any well-typed C_x, C_y, and C_z, if $C_x \to C_y$ and $C_x \to C_z$ then $C_y \overset{\sim}{\downarrow} C_z$.*

Proof Sketch. Case analysis on the two reductions:

Case Both reductions are to-data reductions: The choice of to-data reduction is deterministic, so $C_y = C_z$.

Case Both reductions are task reductions: Case analysis on the task reductions, there are three cases: The choice of task reduction at a given station is deterministic. C_u can be constructed such that $C_y \to C_u$ and C_v such that $C_z \to C_v$ (each reduction replicates the task reduction performed in the other first step).

Case 5: One reduction is a frontend or to-data reduction and the other is a task reduction Since the choice of frontend and to-data reduction is deterministic. □

Lemma 2 (Local Confluence Modulo $\sim$ (P2))**.** *For any well-typed C_x, C_y, and C_z, if $C_x \sim C_z$ and $C_x \to C_y$ then $C_y \overset{\sim}{\downarrow} C_z$.*

Proof Sketch. Case analysis on the reduction:

Case to-data reduction: Replicate the reduction since frontend and to-data reductions do not interfere with any optimized operations.

Case task reduction: Replicate task reduction. □

P1 and P2 for local confluence modulo $\sim$ are illustrated in Fig. 7

Definition 3 (A Normal Form)**.** *We say C' is a normal form of C if $C \to^* C'$ and $C' = \langle \mathbb{D}; []; \{\}; v\rangle$.*

$$\begin{array}{rcll}\mathbb{B} & ::= & \langle \bullet; O; R\rangle & \textit{backend context}\\ \mathbb{T} & ::= & \mathbb{B}[D \mathbin{++} \bullet \mathbin{++} D] & \textit{task context}\end{array}$$

Figure 2: Evaluation Contexts

$$\textsc{Update}\ \frac{N_i = \mathbf{N}\langle k; v_i\rangle \text{ for } i = 1,2 \qquad f(v_1) = v_2}{\mathbb{T}[\langle N_1; [\ell \mapsto \texttt{update}\ f\ k] :: O\rangle] \to \mathbb{T}[\langle N_2; O\rangle] \lhd [\ell \mapsto 0]}$$

$$\textsc{Query}\ \frac{N = \mathbf{N}\langle k; v\rangle}{\mathbb{T}[\langle N; [\ell \mapsto \texttt{query}\ f\ k] :: O\rangle] \to \mathbb{T}[\langle N; O\rangle] \lhd [\ell \mapsto f(v)]}$$

$$\textsc{Add}\ \frac{k \text{ fresh}}{\langle D; [\ell \mapsto \texttt{add}\ v] :: O; R\rangle \to \langle \mathbf{N}\langle k; v\rangle :: D; O; R\rangle \lhd [\ell \mapsto k]}$$

$$\textsc{Prop}\ \frac{k \notin \bigcup_{o \in ran(U)} \odot(o) \qquad D_i = [\langle \mathbf{N}\langle k; v\rangle; O_i\rangle] \text{ for } i = 1,2 \qquad O_1 = U :: O_2}{\mathbb{T}[D_1 \mathbin{++} D] \to \mathbb{T}[D_2 \mathbin{++} (D \mathbin{\underline{\blacktriangleleft}} U)]}$$

$$\textsc{Last}\ \frac{\mathbb{T} = \mathbb{B}[D \mathbin{++} \bullet] \qquad N = \mathbf{N}\langle k; v\rangle \qquad k \notin \odot(o)}{\mathbb{T}[\langle N; \ell \mapsto o :: O\rangle] \to \mathbb{T}[\langle N; O\rangle] \lhd [\ell \mapsto \top]} \qquad \textsc{Empty}\ \frac{o \neq \texttt{add}\ v}{\langle []; [\ell \mapsto o] :: O; R\rangle \to \langle []; O; R\rangle \lhd [\ell \mapsto \top]}$$

$$\textsc{First}\ \frac{o \neq \texttt{add}\ v}{\langle D; [\ell \mapsto o] :: O; R\rangle \to \langle D; O; R\rangle \mathbin{\underline{\blacktriangleleft}} [\ell \mapsto o]}$$

Figure 3: DON Calculus Operational Semantics

Lemma 3 (Confluence Modulo $\sim$)**.** *For any well-typed C_x, C'_x, C_y, and C'_y, if $C_x \sim C_y$, and $C_x \to^* C'_x$, and $C_y \to^* C'_y$, then there exists $\bar{C}_x$ and $\bar{C}_y$ such that $C'_x \to^* \bar{C}_x$, and $C'_y \to^* \bar{C}_y$, and $\bar{C}_x \sim \bar{C}_y$.*

Proof Sketch. Induction on $\to$ and the following property:

$$P(C_x, C_y) : C_x \sim C_y \implies [\forall C'_x, C'_y. C_x \to^* C'_x \wedge C_y \to^* C'_y \implies C'_x \overset{\sim}{\downarrow} C'_y]$$

Given C_x, C_y, C'_x, and C'_y, where $C_x \sim C_y$, and $C_x \xrightarrow{n} C'_x$ and $C_y \xrightarrow{m} C'_y$, we must show that there exists $\bar{C}_x$ and $\bar{C}_y$ such that $C'_x \to^* \bar{C}_x$, and $C'_y \to^* \bar{C}_y$, and $\bar{C}_x \sim \bar{C}_y$.

If $n = 0$ and $m = 0$ then the result directly follows. Otherwise, assume $n > 0$ such that $C_x \to C_{x_1} \to^* C'_x$. Applying P2 to C_x, C_y, and C_{x_1}, we get C_u and C_v such that $C_{x_1} \to^* C_u$, and $C_y \to^* C_v$, and $C_u \sim C_v$. There are two cases:

Case 1 $m = 0$. Let $\bar{C}'_x$ and $\bar{C}_u$ and $\bar{C}_v$ be normal forms of C'_x, C_u, and C_v. $\bar{C}'_x \sim \bar{C}_u$ by the induction hypothesis $P(C_{x_1}, C_{x_1})$, and $\bar{C}_u \sim \bar{C}_v$ by the induction hypothesis $P(C_u, C_v)$, which completes the proof. The diagram for this case is shown in Fig. 8.

Case 2 $m > 0$. Assume $C_y \to C_{y_1} \to^* C'_y$. There are two sub cases:

Case 2a $C_v = C_y$. Applying P2 to C_y, C_u, and C_{y_1}, we get C_w and C_z such that $C_u \to^* C_w$, and $C_{y_1} \to^* C_z$, and $C_w \sim C_z$. Let $\bar{C}'_x$, $\bar{C}_w$, and $\bar{C}_z$ be normal forms of C'_x, C_w, and C_z, respectively. $\bar{C}'_x \sim \bar{C}_w$ by the induction hypothesis $P(C_{x_1}, C_{x_1})$, and $\bar{C}_w \sim \bar{C}_z$ by $P(C_w, C_z)$, and $\bar{C}_z \sim \bar{C}'_y$ by $P(C_{y_1}, C_{y_1})$, which completes the proof. The diagram for this case is shown in Fig. **??**.

Case 2b Otherwise, Assume $C_y \to C_t \to^* C_v$. Applying P1 to C_y, C_{y_1}, and C_t, we get C_w and C_z such that $C_u \to^* C_w$, and $C_{y_1} \to^* C_z$, and $C_w \sim C_z$. Let $\bar{C}'_x$, $\bar{C}_u$, $\bar{C}_v$, $\bar{C}_w$, $\bar{C}_z$, and $\bar{C}'_y$ be normal forms of C'_x, C_u, C_v, C_w, C_z, and C'_y, respectively. $\bar{C}'_x \sim \bar{C}_u$ by the induction hypothesis $P(C_{x_1}, C_{x_1})$, and $\bar{C}_u \sim \bar{C}_v$ by $P(C_u, C_v)$, and $\bar{C}_v \sim \bar{C}_w$ by $P(C_t, C_t)$, and $\bar{C}_w \sim \bar{C}_z$ by $P(C_w, C_z)$, and $\bar{C}_z \sim \bar{C}'_y$ by $P(C_{y_1}, C_{y_1})$, which completes the proof. The diagram for this case is shown in Fig. 11(b). □

Definition 4 (Final Configuration)**.** *$\langle D; O; R\rangle$ is a final configuration iff $O = []$ and $D = \overline{\langle N; []\rangle}$ for some $\overline{N}$.*

Definition 5 (Convergent Configurations)**.** *We say two configurations $\langle D_1; O_1; R_1\rangle$ and $\langle D_2; O_2; R_2\rangle$ are convergent configurations iff (1) $D_1 = D_2$ and (2) $dom(R_1) = dom(R_2)$ and $\forall \ell \in dom(R_1). R_1(\ell) = R_2(\ell)$.*

Theorem 1 (Determinism)**.** *For any operations O and data D, if $\langle D; O; []\rangle \to^* C_1$ and $\langle D; O; []\rangle \to^* C_2$ and C_1 and C_2 are final, then C_1 and C_2 are convergent.*

According to this theorem, all terminating executions not only produce the same results for operations, but also lead to the same final data structure.

Corollary 1 (Sequential Consistency)**.** *For any operations $O = \overline{\ell \mapsto o}^m$ and data D and (1) $\langle D; O; []\rangle \to^* C_1$ where C_1 is final and (2)*

$$\frac{D \overset{b}{\sim} D'}{\langle D;O;R\rangle \sim \langle D';O;R\rangle} \qquad [] \overset{b}{\sim} [] \qquad \frac{D \overset{b}{\sim} D' \qquad \forall k'.\mathit{writes}(\mathit{fl}(O),k') \equiv \mathit{writes}(\mathit{fl}(O'),k') \qquad \mathit{reads}(\mathit{fl}(O)) \overset{\cup}{\equiv} \mathit{reads}(\mathit{fl}(O'))}{\langle\langle k;v\rangle;O\rangle :: D \overset{b}{\sim} \langle\langle k;v\rangle;O'\rangle :: D'}$$

$$\mathit{writes}([],k) \triangleq \mathtt{Id} \qquad \frac{}{\mathit{writes}(O +\!\!+ [\ell \mapsto \mathtt{update}\ f\ k],k) \triangleq f \circ \mathit{writes}(O,k)} \qquad \frac{k \neq \odot(o) \vee o \neq \mathtt{update}\ f\ k' \text{ for some } f\ k'}{\mathit{writes}(O +\!\!+ [\ell \mapsto o],k) \triangleq \mathit{writes}(O,k)}$$

$$\mathit{reads}([]) \triangleq \{\} \qquad \mathit{reads}(O +\!\!+ [\ell \mapsto \mathtt{query}\ f\ k]) \triangleq \{\ell \mapsto f :: \mathit{writes}(O,k)\} \cup \mathit{reads}(O,k)$$

$$\mathit{reads}(O +\!\!+ [\ell \mapsto \mathtt{update}\ f\ k]) \triangleq \mathit{reads}(O)$$

Figure 4: Configuration Equivalence: $C \sim C'$

$$\begin{tikzcd} & C_x \arrow[dl] \arrow[dr] & \\ C_y \arrow[dr, dashed, "*"] & & C_z \arrow[dl, dashed, "*"'] \\ & C_u{\sim}C_v & \end{tikzcd}$$

Figure 5: P1

$$\begin{tikzcd} & C_x{\sim}C_z \arrow[dl] \arrow[dd, dashed, "*"] \\ C_y \arrow[dr, dashed, "*"] & \\ & C_u{\sim}C_v \end{tikzcd}$$

Figure 6: P2

Figure 7: Two Properties of Local Confluence Modulo $\sim$

$$\begin{array}{rcl} \langle D;[\ell_1 \mapsto o_1];[]\rangle & \to^* & \langle D_{2_1};[];R_{2_1}\rangle \\ \langle D_{2_1};[\ell_2 \mapsto o_2];[]\rangle & \to^* & \langle D_{2_2};[];R_{2_2}\rangle \\ & \cdots & \\ \langle D_{2_{m-1}};[\ell_m \mapsto o_m];[]\rangle & \to^* & \langle D_{2_m};[];R_{2_m}\rangle \end{array}$$

where each post-reduction configuration is final, then C_1 *and* $\langle D_{2_m};[];R_{2_1} \uplus \cdots \uplus R_{2_m}\rangle$ *are convergent.*

Proof Sketch. Immediately follows Theorem. 1. Note that configuration $\langle D_{2_m};[];R_{2_1} \uplus \cdots \uplus R_{2_m}\rangle$ is formed through a concrete order of reduction of the more general form. Since Theorem. 1 establishes that any two arbitrary reductions lead to the same result, it will subsume this concrete order of reduction. □

4. Experimental Data in Perpetual UP/OP Scenarios

To complete the design space exploration, we also constructed experiments when the system is perpetually in the UP state, and perpetually in the OP state. For the former, the time between two operation arrivals is randomly and uniformly set among [0, 10ms], and in the latter, among [290, 300ms].

To summarize, (1) PITSTOP outperforms the two baselines significantly in the case of perpetual UP, in that the cascaded effect is "perpetually" incurred, and amplified as time goes on; (2) PITSTOP is not as efficient as MP in the scenario of perpetual OP, even though more efficient than BMP. In other words, when operations arrive at a very slow rate, the data processing engine degenerates to *offline* processing, and any feature specifically targeting *online* optimization — either PITSTOP or BMP — only adds to the overhead without benefit over the minimalistic MP design. Overall, perpetual UP/OP are less interesting use scenarios because they are symptoms of suboptimal system configuration and administration: more/less computational resources should be configured.

5. Experimental Data Over Different LDSIZE Values

Finally, we show the impact of `LDSIZE` for latency (Fig. 14) and throughput (Fig. 15).

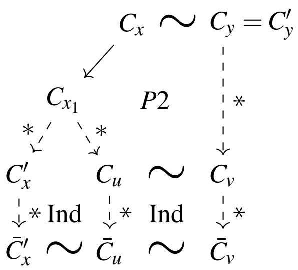


Figure 8: Case 1

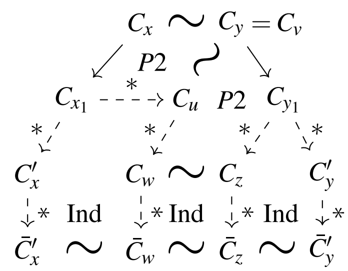


Figure 9: Case 2a

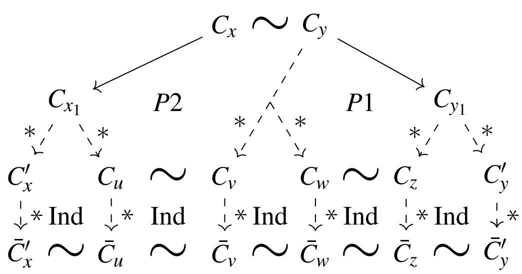
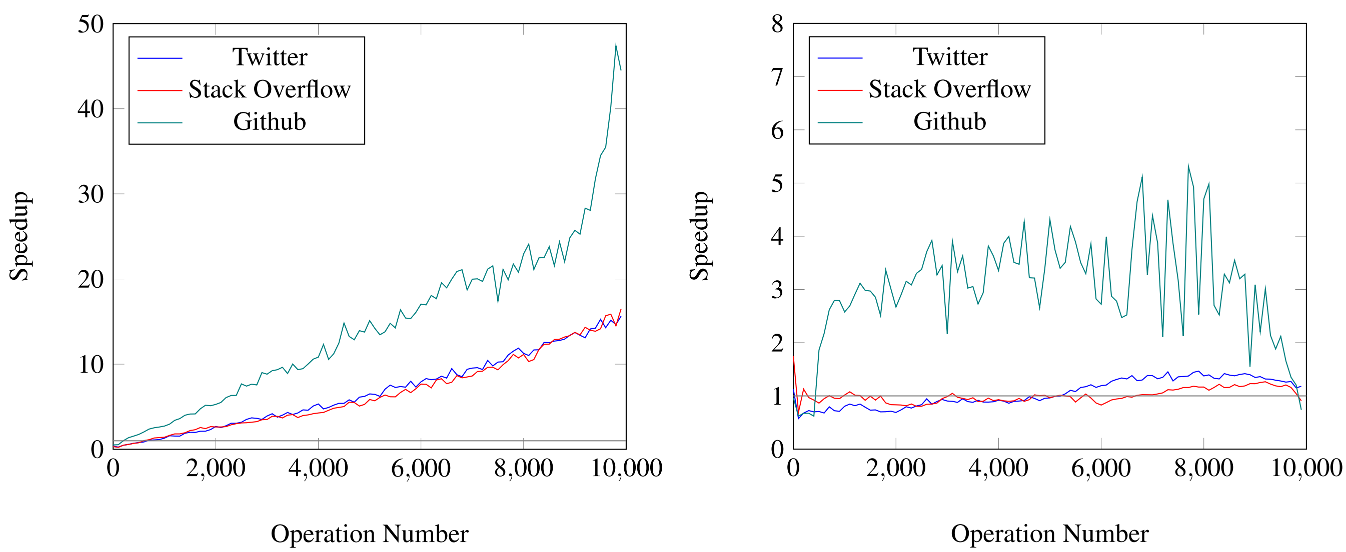


Figure 10: Case 2b

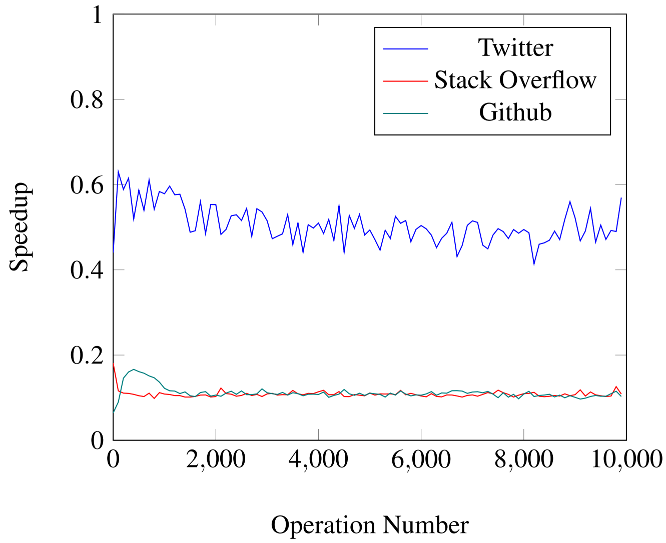
Figure 11: Cases for Confluence



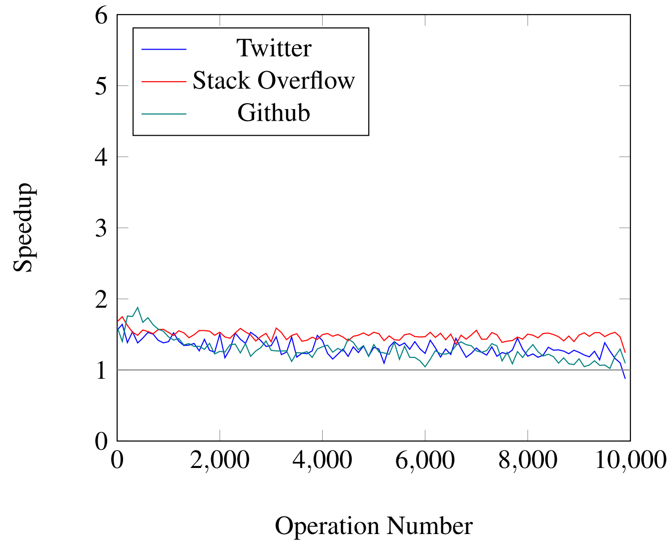
(a) over MP

(b) over BMP

Figure 12: PITSTOP Latency Speedup in Perpetual UP Scenarios (higher is better)

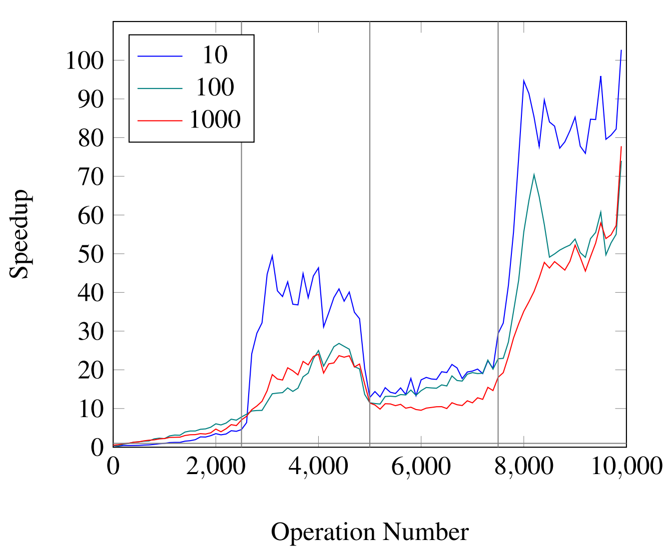


(a) Over MP

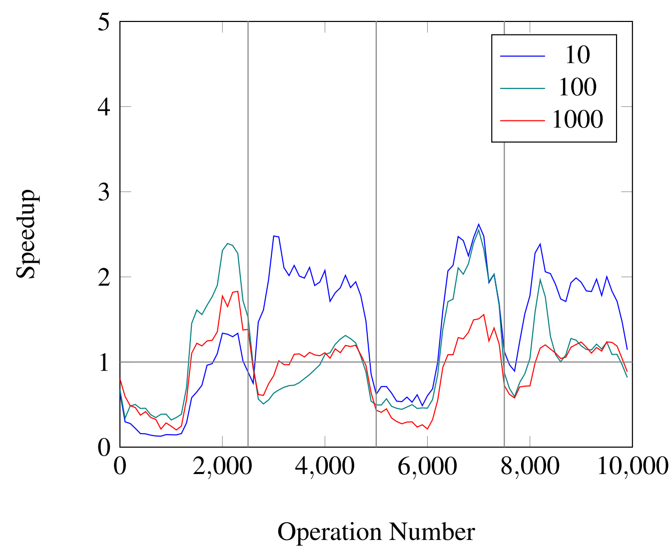


(b) Over BMP

Figure 13: PITSTOP Latency Speedup in Perpetual OP Scenarios (higher is better)

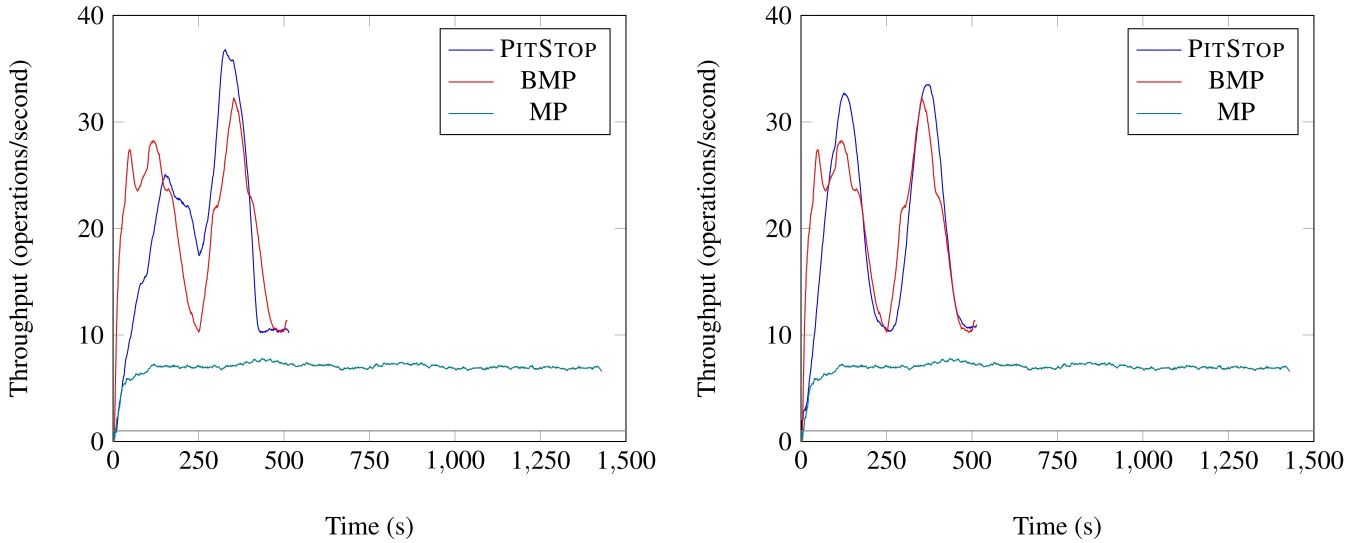


(a) Latency speedup of PITSTOP over MP.



(b) Latency speedup of PITSTOP over BMP.

Figure 14: Latency Speedup for `F1` for different *LDSIZE* for Twitter



(a) *LDSIZE* $= 10$

(b) *LDSIZE* $= 1000$

Figure 15: Rolling throughput for the Twitter dataset in `F1` scenario for different *LDSIZE* with window size as 100s