国立研究開発法人理化学研究所　御中

VSD-R21001-00

Society5.0 シミュレーションの高信頼実行のための チェックポイント・リスタート・シミュレーターの開発

作業報告書（暫定版）

2021 年 2 月

株式会社 ヴァイナス

目次

1 受託業務件名

Society5.0 シミュレーションの高信頼実行のためのチェックポイント・リスタート・シミュレーターの開発

2 はじめに

本件業務では、Society5.0 シミュレーションの高信頼実行のためのチェックポイント・リスタート・シミュレーターを実現する関数の開発を実施しました。本報告書では、その開発業務内容について報告します。

3 開発概要

以下の2つの関数を実装しました。

(ア) simulate_cr 関数:
マルチレベル C/R をシミュレーションし、その結果を出力します。

(イ) optimize_cr 関数:
simulate_cr 関数の efficiency を最大化する、interval、L2_freq を見つけ出し、その時のシミュレーション結果を出力します。

4 開発詳細

4.1 開発環境

各関数の開発は、以下の環境で行いました。
OS: Windows 10
言語: Python 3.7.4

4.2 simulate_cr 関数

以下の通り、simulate_cr 関数の開発を行いました。

tuple simulate_cr(interval, L2ckpt_freq, L1ckpt_overhead, L2ckpt_latency, ckptRestartTimes, failRates, N, SN, G, g, alpha, check_interval, n_check_ok, n_failure_max, efficiency_log)

・引数（入力）

引数名	説明	型	
Interval	L1 チェックポイントの間隔	int	必須
L2ckpt_freq	L2 チェックポイントの頻度	int	必須
L1ckpt_overhead	同期 L1 チェックポイントの時間	int	必須
L2ckpt_latency	非同期 L2 チェックポイントの時間	int	必須
ckptRestartTimes	L1,L2 リカバリに要する時間を格納した長さ 2 の配列 = [L1 リカバリ時間,L2 リカバリ時間]	List [int,int]	必須
failRates	L1,L2 リカバリを必要とする障害の単位時間に発生する回数を格納した長さ 2 の配列 = [L1 障害回数,L2 障害回数]	List [float,float]	必須
N	全計算ノード数	int	必須
SN	予備ノード数 ※仕様追加されたパラメータ	int	必須
G	L1 チェックポイントのグループサイズ	int	必須
g	L1 チェックポイントの障害耐性	int	必須
alpha	シミュレーション終了とする Efficiency の変化量の閾値	float	必須
check_interval	Efficiency の変化量チェックの頻度 ※仕様追加されたパラメータ	int	省略可、Default=1
n_check_ok	Efficiency の変化量チェックで終了と判定されるための連続回数 ※仕様追加されたパラメータ	int	省略可、Default=1
n_failure_max	最大障害発生回数 ※仕様追加されたパラメータ	int	省略可、Default=500000
efficiency_log	Efficiency 変化量チェックの履歴出力のオン・オフ ※仕様追加されたパラメータ	bool	省略可、Default=False

返り値（出力）：tuple 型データ=(X,A,B,C,D,E,F)

引数名	説明	型
X	Efficiency = A/(B+C+D+F)	float
A	実質計算時間	float
B	計算状態に費やした時間	float
C	L1 チェックポイントに費やした時間	float
D	L1 リカバリに費やした時間	float
E	L2 チェックポイントに費やした時間	float
F	L2 リカバリに費やした時間	float

4.3 optimize_cr 関数

以下の通り、optimize_cr 関数の開発を行いました。

tuple optimize_cr (L1ckpt_overhead, L2ckpt_latency, ckptRestartTimes, failRates, N, SN, G, g, alpha, check_interval, n_check_ok, n_failure_max, n_steps, log_interval)

・引数（入力）

引数名	説明	型	
L1ckpt_overhead	同期 L1 チェックポイントの時間	int	必須
L2ckpt_latency	非同期 L2 チェックポイントの時間	int	必須
ckptRestartTimes	L1,L2 リカバリに要する時間を格納した長さ 2 の配列 = [L1 リカバリ時間,L2 リカバリ時間]	List [int,int]	必須
failRates	L1,L2 リカバリを必要とする障害の単位時間に発生する回数を格納した長さ 2 の配列 = [L1 障害回数,L2 障害回数]	List [float,float]	必須
N	全計算ノード数	int	必須
SN	予備ノード数 ※仕様追加されたパラメータ	int	必須
G	L1 チェックポイントのグループサイズ	int	必須
g	L1 チェックポイントの障害耐性	int	必須
alpha	シミュレーション終了とする Efficiency の変化量の閾値	float	必須
check_interval	Efficiency の変化量チェックの頻度 ※仕様追加されたパラメータ	int	省略可、Default=1
n_check_ok	Efficiency の変化量チェックで終了と判定されるための連続回数 ※仕様追加されたパラメータ	int	省略可、Default=1
n_failure_max	最大障害発生回数 ※仕様追加されたパラメータ	int	省略可、Default=500000
n_steps	最適化の反復回数※仕様追加されたパラメータ	int	省略可、Default=5000
log_interval	最適化のログ出力間隔、0 とすると出力なし ※仕様追加されたパラメータ	int	省略可、Default=100

・返り値（出力）：tuple 型データ=(X,A,B,C,D,E,F, interval, L2ckpt_freq)

引数名	説明	型
X	最適化結果の interval, L2ckpt_freq 時の Efficiency = A/(B+C+D+F)	float
A	最適化結果の interval, L2ckpt_freq 時の実質計算時間	float
B	最適化結果の interval, L2ckpt_freq 時の計算状態に費やした時間	float
C	最適化結果の interval, L2ckpt_freq 時の L1 チェックポイントに費やした時間	float
D	最適化結果の interval, L2ckpt_freq 時の L1 リカバリに費やした時間	float

E	最適化結果の interval, L2ckpt_freq 時の L2 チェックポイントに費やした時間	float
F	最適化結果の interval, L2ckpt_freq 時の L2 リカバリに費やした時間	float
interval	最適化結果の L1 チェックポイントの間隔	int
L2ckpt_freq	最適化結果の L2 チェックポイントの頻度	int

◆最適化手法について

最適化手法には、焼きなまし法を採用しました。

・初期状態

下記の interval、L2_freq_freq 組み合わせ（24 通り）の内、最も Efficiency の高いものを初期状態とするよう実装しました。

interval = 1000, 2500, 5000, 8000, 12000, 24000

L2_freq_freq = 1, 2, 5, 10

・状態遷移

状態遷移については、下記の4つの方法を検討しました。

方法 1:

1. interval と L2ckpt_freq のどちらの数値を変えるかをランダムに選択
2. 選択されたパラメータを 2%増減

方法 2:

1. interval と L2ckpt_freq のどちらの数値を変えるかをランダムに選択
2. 選択されたパラメータを 5%以内のランダムな値で増減

方法 3:

1. interval と L2ckpt_freq の両方を 0～5%以内のランダムな値で増減

方法 4:

1. interval と L2ckpt_freq のどちらの数値を変えるかをランダムに選択
2. 選択されたパラメータを固定値で増減

検討の結果、方法 4（※）以外は、どれもあまり差が見られなかったため、方法 1 を採用しています。

※interval は範囲が広いため、固定値で増減する場合、小さい値にすると範囲内の移動に回数が掛かりすぎ、大きい値にすると小さい側で変化量が大きくなりすぎる問題が発生しました。

上記の状態遷移の方法は、簡単なソースコード修正で、上記いずれの方法にも変更できるようにしていますので、必要に応じて修正してご利用ください。また、2%や 5%の数字もソースコードの対応箇所の変更のみで変更可能です。

5 動作確認テスト結果

5.1 テスト環境

各関数のテストは、以下の環境で行いました。

Windows： Windows 10 + Python 3.7.4

Linux： Ubuntu 16.04 + Python 3.5.2

5.2 simulate_cr 関数

simulate_cr 関数の動作確認結果は以下の通りで、問題なく動作しております。

区分	テスト内容	結果確認内容	判定	備考
実行	シミュレーションの動作確認 （デバッグ出力での処理確認）	シミュレーション（デバッグ出力内容）が想定した処理、フローとなっていること	OK	デバッグ出力オン
設定範囲	interval=0	シミュレーションが正常に終了すること Efficiency=1.0 となっていること CUI に特殊設定の注記メッセージが表示されること	OK	L2_ckpt_freq=0
設定範囲	interval=1	シミュレーションが正常に終了すること	OK	
設定範囲	interval=2		OK	
設定範囲	interval=3		OK	
設定範囲	interval=4		OK	
設定範囲	interval=5		OK	
設定範囲	interval=6		OK	
設定範囲	interval=7		OK	
設定範囲	interval=8		OK	
設定範囲	interval=9		OK	
設定範囲	interval=10		OK	
設定範囲	interval=20		OK	
設定範囲	interval=30		OK	
設定範囲	interval=40		OK	
設定範囲	interval=50		OK	
設定範囲	interval=60		OK	
設定範囲	interval=70		OK	
設定範囲	interval=80		OK	
設定範囲	interval=90		OK	
設定範囲	interval=100		OK	
設定範囲	interval=200		OK	
設定範囲	interval=300		OK	

設定範囲	interval=400		OK	
設定範囲	interval=500		OK	
設定範囲	interval=600		OK	
設定範囲	interval=700		OK	
設定範囲	interval=800		OK	
設定範囲	interval=900		OK	
設定範囲	interval=1000		OK	
設定範囲	interval=2000		OK	
設定範囲	interval=3600		OK	
設定範囲	interval=7200		OK	
設定範囲	interval=10000		OK	
設定範囲	interval=18000		OK	
設定範囲	interval=36000		OK	
設定範囲	L2ckpt_freq=0	シミュレーションが正常に終了すること CUI に特殊設定の注記メッセージが表示されること	OK	L2_ckpt_freq=0
設定範囲	L2ckpt_freq=1	シミュレーションが正常に終了すること	OK	
設定範囲	L2ckpt_freq=2		OK	
設定範囲	L2ckpt_freq=3		OK	
設定範囲	L2ckpt_freq=4		OK	
設定範囲	L2ckpt_freq=5		OK	
設定範囲	L2ckpt_freq=6		OK	
設定範囲	L2ckpt_freq=7		OK	
設定範囲	L2ckpt_freq=8		OK	
設定範囲	L2ckpt_freq=9		OK	
設定範囲	L2ckpt_freq=10		OK	
設定範囲	L2ckpt_freq=20		OK	
設定範囲	L2ckpt_freq=30		OK	
設定範囲	L2ckpt_freq=40		OK	
設定範囲	L2ckpt_freq=50		OK	
設定範囲	L2ckpt_freq=60		OK	
設定範囲	L2ckpt_freq=70		OK	
設定範囲	L2ckpt_freq=80		OK	
設定範囲	L2ckpt_freq=90		OK	
設定範囲	L2ckpt_freq=100		OK	
設定範囲	L1ckpt_overhead=0	シミュレーションが正常に終了すること	OK	
設定範囲	L1ckpt_overhead=1		OK	
設定範囲	L1ckpt_overhead=2		OK	
設定範囲	L1ckpt_overhead=3		OK	

設定範囲	L1ckpt_overhead=4		OK	
設定範囲	L1ckpt_overhead=5		OK	
設定範囲	L1ckpt_overhead=6		OK	
設定範囲	L1ckpt_overhead=7		OK	
設定範囲	L1ckpt_overhead=8		OK	
設定範囲	L1ckpt_overhead=9		OK	
設定範囲	L1ckpt_overhead=10		OK	
設定範囲	L1ckpt_overhead=20		OK	
設定範囲	L1ckpt_overhead=30		OK	
設定範囲	L1ckpt_overhead=40		OK	
設定範囲	L1ckpt_overhead=50		OK	
設定範囲	L1ckpt_overhead=60		OK	
設定範囲	L1ckpt_overhead=70		OK	
設定範囲	L1ckpt_overhead=80		OK	
設定範囲	L1ckpt_overhead=90		OK	
設定範囲	L1ckpt_overhead=100		OK	
設定範囲	L1ckpt_overhead=200		OK	
設定範囲	L1ckpt_overhead=300		OK	
設定範囲	L1ckpt_overhead=400		OK	
設定範囲	L1ckpt_overhead=500		OK	
設定範囲	L1ckpt_overhead=600		OK	
設定範囲	L1ckpt_overhead=700		OK	
設定範囲	L1ckpt_overhead=800		OK	
設定範囲	L1ckpt_overhead=900		OK	
設定範囲	L1ckpt_overhead=1000		OK	
設定範囲	L1ckpt_overhead=1800		OK	
設定範囲	L1ckpt_overhead=3600		OK	
設定範囲	L2ckpt_latency=0	シミュレーションが正常に終了すること	OK	
設定範囲	L2ckpt_latency=1		OK	
設定範囲	L2ckpt_latency=2		OK	
設定範囲	L2ckpt_latency=3		OK	
設定範囲	L2ckpt_latency=4		OK	
設定範囲	L2ckpt_latency=5		OK	
設定範囲	L2ckpt_latency=6		OK	
設定範囲	L2ckpt_latency=7		OK	
設定範囲	L2ckpt_latency=8		OK	
設定範囲	L2ckpt_latency=9		OK	
設定範囲	L2ckpt_latency=10		OK	

設定範囲	L2ckpt_latency=20		OK	
設定範囲	L2ckpt_latency=30		OK	
設定範囲	L2ckpt_latency=40		OK	
設定範囲	L2ckpt_latency=50		OK	
設定範囲	L2ckpt_latency=60		OK	
設定範囲	L2ckpt_latency=70		OK	
設定範囲	L2ckpt_latency=80		OK	
設定範囲	L2ckpt_latency=90		OK	
設定範囲	L2ckpt_latency=100		OK	
設定範囲	L2ckpt_latency=200		OK	
設定範囲	L2ckpt_latency=300		OK	
設定範囲	L2ckpt_latency=400		OK	
設定範囲	L2ckpt_latency=500		OK	
設定範囲	L2ckpt_latency=600		OK	
設定範囲	L2ckpt_latency=700		OK	
設定範囲	L2ckpt_latency=800		OK	
設定範囲	L2ckpt_latency=900		OK	
設定範囲	L2ckpt_latency=1000		OK	
設定範囲	L2ckpt_latency=2000		OK	
設定範囲	L2ckpt_latency=3600		OK	
設定範囲	L2ckpt_latency=7200		OK	
設定範囲	L2ckpt_latency=10000		OK	
設定範囲	L2ckpt_latency=18000		OK	
設定範囲	L2ckpt_latency=36000		OK	
設定範囲	ckptRestartTimes[0]=0	シミュレーションが正常に終了すること	OK	
設定範囲	ckptRestartTimes[0]=1		OK	
設定範囲	ckptRestartTimes[0]=2		OK	
設定範囲	ckptRestartTimes[0]=3		OK	
設定範囲	ckptRestartTimes[0]=4		OK	
設定範囲	ckptRestartTimes[0]=5		OK	
設定範囲	ckptRestartTimes[0]=6		OK	
設定範囲	ckptRestartTimes[0]=7		OK	
設定範囲	ckptRestartTimes[0]=8		OK	
設定範囲	ckptRestartTimes[0]=9		OK	
設定範囲	ckptRestartTimes[0]=10		OK	
設定範囲	ckptRestartTimes[0]=20		OK	
設定範囲	ckptRestartTimes[0]=30		OK	
設定範囲	ckptRestartTimes[0]=40		OK	

設定範囲	ckptRestartTimes[0]=50		OK	
設定範囲	ckptRestartTimes[0]=60		OK	
設定範囲	ckptRestartTimes[0]=70		OK	
設定範囲	ckptRestartTimes[0]=80		OK	
設定範囲	ckptRestartTimes[0]=90		OK	
設定範囲	ckptRestartTimes[0]=100		OK	
設定範囲	ckptRestartTimes[0]=200		OK	
設定範囲	ckptRestartTimes[0]=300		OK	
設定範囲	ckptRestartTimes[0]=400		OK	
設定範囲	ckptRestartTimes[0]=500		OK	
設定範囲	ckptRestartTimes[0]=600		OK	
設定範囲	ckptRestartTimes[0]=700		OK	
設定範囲	ckptRestartTimes[0]=800		OK	
設定範囲	ckptRestartTimes[0]=900		OK	
設定範囲	ckptRestartTimes[0]=1000		OK	
設定範囲	ckptRestartTimes[0]=1800		OK	
設定範囲	ckptRestartTimes[0]=3600		OK	
設定範囲	ckptRestartTimes[1]=0	シミュレーションが正常に終了すること	OK	
設定範囲	ckptRestartTimes[1]=1		OK	
設定範囲	ckptRestartTimes[1]=2		OK	
設定範囲	ckptRestartTimes[1]=3		OK	
設定範囲	ckptRestartTimes[1]=4		OK	
設定範囲	ckptRestartTimes[1]=5		OK	
設定範囲	ckptRestartTimes[1]=6		OK	
設定範囲	ckptRestartTimes[1]=7		OK	
設定範囲	ckptRestartTimes[1]=8		OK	
設定範囲	ckptRestartTimes[1]=9		OK	
設定範囲	ckptRestartTimes[1]=10		OK	
設定範囲	ckptRestartTimes[1]=20		OK	
設定範囲	ckptRestartTimes[1]=30		OK	
設定範囲	ckptRestartTimes[1]=40		OK	
設定範囲	ckptRestartTimes[1]=50		OK	
設定範囲	ckptRestartTimes[1]=60		OK	
設定範囲	ckptRestartTimes[1]=70		OK	
設定範囲	ckptRestartTimes[1]=80		OK	
設定範囲	ckptRestartTimes[1]=90		OK	
設定範囲	ckptRestartTimes[1]=100		OK	
設定範囲	ckptRestartTimes[1]=200		OK	

設定範囲	ckptRestartTimes[1]=300		OK	
設定範囲	ckptRestartTimes[1]=400		OK	
設定範囲	ckptRestartTimes[1]=500		OK	
設定範囲	ckptRestartTimes[1]=600		OK	
設定範囲	ckptRestartTimes[1]=700		OK	
設定範囲	ckptRestartTimes[1]=800		OK	
設定範囲	ckptRestartTimes[1]=900		OK	
設定範囲	ckptRestartTimes[1]=1000		OK	
設定範囲	ckptRestartTimes[1]=2000		OK	
設定範囲	ckptRestartTimes[1]=3600		OK	
設定範囲	ckptRestartTimes[1]=7200		OK	
設定範囲	ckptRestartTimes[1]=10000		OK	
設定範囲	ckptRestartTimes[1]=18000		OK	
設定範囲	ckptRestartTimes[1]=36000		OK	
設定範囲	failRates[0]=0.0	シミュレーションが正常に終了すること	OK	
設定範囲	failRates[0]=1.0×10^{-10}		OK	
設定範囲	failRates[0]=1.0×10^{-9}		OK	
設定範囲	failRates[0]=1.0×10^{-8}		OK	
設定範囲	failRates[0]=1.0×10^{-7}		OK	
設定範囲	failRates[0]=1.0×10^{-6}		OK	
設定範囲	failRates[0]=1.0×10^{-5}		OK	
設定範囲	failRates[0]=1.0×10^{-4}		OK	
設定範囲	failRates[0]=1.0×10^{-3}		OK	
設定範囲	failRates[0]=1.0×10^{-2}		OK	
設定範囲	failRates[0]=1.0×10^{-1}		OK	
設定範囲	failRates[0]=1.0		OK	
設定範囲	failRates[1]=0.0	シミュレーションが正常に終了すること	OK	
設定範囲	failRates[1]=1.0×10^{-10}		OK	
設定範囲	failRates[1]=1.0×10^{-9}		OK	
設定範囲	failRates[1]=1.0×10^{-8}		OK	
設定範囲	failRates[1]=1.0×10^{-7}		OK	
設定範囲	failRates[1]=1.0×10^{-6}		OK	
設定範囲	failRates[1]=1.0×10^{-5}		OK	
設定範囲	failRates[1]=1.0×10^{-4}		OK	
設定範囲	failRates[1]=1.0×10^{-3}		OK	
設定範囲	failRates[1]=1.0×10^{-2}		OK	
設定範囲	failRates[1]=1.0×10^{-1}		OK	
設定範囲	failRates[1]=1.0		OK	

設定範囲	failRates=[0.0,0.0]	シミュレーションが正常に終了すること Efficiency=interval/(interval+L1ckpt_overhead)となること CUI に特殊設定の注記メッセージが表示されること	OK	L1、L2 の障害回数が共に 0 の特殊なケース
設定範囲	N=1	シミュレーションが正常に終了すること	OK	
設定範囲	N=2		OK	
設定範囲	N=3		OK	
設定範囲	N=4		OK	
設定範囲	N=5		OK	
設定範囲	N=6		OK	
設定範囲	N=7		OK	
設定範囲	N=8		OK	
設定範囲	N=9		OK	
設定範囲	N=10		OK	
設定範囲	N=20		OK	
設定範囲	N=50		OK	
設定範囲	N=100		OK	
設定範囲	N=200		OK	
設定範囲	N=500		OK	
設定範囲	N=1000		OK	
設定範囲	N=2000		OK	
設定範囲	N=5000		OK	
設定範囲	N=10000		OK	
設定範囲	N=20000		OK	
設定範囲	N=50000		OK	
設定範囲	N=100000		OK	
設定範囲	N=200000		OK	
設定範囲	N=500000		OK	
設定範囲	N=1000000		OK	
設定範囲	G=1	シミュレーションが正常に終了すること	OK	
設定範囲	G=2		OK	
設定範囲	G=3		OK	
設定範囲	G=4		OK	
設定範囲	G=5		OK	
設定範囲	G=6		OK	
設定範囲	G=7		OK	
設定範囲	G=8		OK	
設定範囲	G=9		OK	

設定範囲	G=10		OK	
設定範囲	G=20		OK	
設定範囲	G=50		OK	
設定範囲	G=100		OK	
設定範囲	G=200		OK	
設定範囲	G=500		OK	
設定範囲	G=1000		OK	
設定範囲	G=2000		OK	
設定範囲	G=5000		OK	
設定範囲	G=10000		OK	
設定範囲	G=20000		OK	
設定範囲	G=50000		OK	
設定範囲	G=100000		OK	
設定範囲	G=200000		OK	
設定範囲	G=500000		OK	
設定範囲	G=1000000		OK	
設定範囲	g=1	シミュレーションが正常に終了すること	OK	
設定範囲	g=2		OK	
設定範囲	g=3		OK	
設定範囲	g=4		OK	
設定範囲	g=5		OK	
設定範囲	g=6		OK	
設定範囲	g=7		OK	
設定範囲	g=8		OK	
設定範囲	g=9		OK	
設定範囲	g=10		OK	
設定範囲	g=20		OK	
設定範囲	g=50		OK	
設定範囲	g=100		OK	
設定範囲	g=200		OK	
設定範囲	g=500		OK	
設定範囲	g=1000		OK	
設定範囲	g=2000		OK	
設定範囲	g=5000		OK	
設定範囲	g=10000		OK	
設定範囲	g=20000		OK	
設定範囲	g=50000		OK	
設定範囲	g=100000		OK	

設定範囲	g=200000		OK	
設定範囲	g=500000		OK	
設定範囲	g=1000000		OK	
設定範囲	alpha=1.0	シミュレーションが正常に終了すること 終了時の Efficiency 変化量が alpha 以下になっていること	OK	
設定範囲	alpha=1.0×10^{-1}		OK	
設定範囲	alpha=1.0×10^{-2}		OK	
設定範囲	alpha=1.0×10^{-3}		OK	
設定範囲	alpha=1.0×10^{-4}		OK	
設定範囲	alpha=1.0×10^{-5}		OK	
設定範囲	alpha=1.0×10^{-6}		OK	
設定範囲	alpha=1.0×10^{-7}		OK	
設定範囲	alpha=1.0×10^{-8}		OK	
設定範囲	alpha=1.0×10^{-9}		OK	
設定範囲	SN=10	シミュレーションが正常に終了すること CUI に予備ノードが不足したこと示す警告メッセージが表示されること Efficiency=0.0 となること	OK	
設定範囲	SN=10000	シミュレーションが正常に終了すること	OK	
設定範囲	SN=1000000		OK	
設定範囲	check_interval=1	シミュレーションが正常に終了すること Efficiency の変化量チェックの頻度が設定通りになっていること	OK	
設定範囲	check_interval=100		OK	
設定範囲	check_interval=1000		OK	
設定範囲	n_check_ok=1	シミュレーションが正常に終了すること 終了時の Efficiency 変化量が設定回数連続で alpha(=1e-3)以下になっていること	OK	
設定範囲	n_check_ok=2		OK	
設定範囲	n_check_ok=3		OK	
設定範囲	n_failure_max=10	シミュレーションが正常に終了すること CUI に障害発生回数が最大値を超えたこと示す警告メッセージが表示されること Efficiency=0.0 となること	OK	
設定範囲	n_failure_max=50000	シミュレーションが正常に終了すること	OK	
設定範囲	efficiency_log=False	シミュレーションが正常に終了すること Efficiency の変化量履歴が出力されないこと	OK	
不正入力	interval=-1	エラーが発生すること CUI に不正入力を示すメッセージが表示されること 返り値がすべて 0 になること	OK	
不正入力	interval="a"		OK	
不正入力	L2ckpt_freq=-1		OK	
不正入力	L2ckpt_freq="a"		OK	
不正入力	L2ckpt_freq=1		OK	他設定に対して小さすぎ
不正入力	L2ckpt_freq=1		OK	interval=0 時は、

	interval=0			L2ckpt_freq=0 のみ有効
不正入力	L1ckpt_overhead=-1		OK	
不正入力	L1ckpt_overhead=”a”		OK	
不正入力	L2ckpt_latency=-1		OK	
不正入力	L2ckpt_latency=”a”		OK	
不正入力	ckptRestartTimes=[-1,2000]		OK	
不正入力	ckptRestartTimes=[”a”,2000]		OK	
不正入力	ckptRestartTimes=[100,-1]		OK	
不正入力	ckptRestartTimes=[100,”a”]		OK	
不正入力	ckptRestartTimes=[100]		OK	
不正入力	ckptRestartTimes=100		OK	
不正入力	failRates=[-1,1e-6]		OK	
不正入力	failRates=[”a”,1e-6]		OK	
不正入力	failRates=[1e-5,-1]		OK	
不正入力	failRates=[1e-5,”a”]		OK	
不正入力	failRates=[1e-5]		OK	
不正入力	failRates=1e-5		OK	
不正入力	N=0		OK	
不正入力	N=”a”		OK	
不正入力	G=0		OK	
不正入力	G=”a”		OK	
不正入力	G=1001		OK	N に対して不適切
不正入力	g=0		OK	
不正入力	g=”a”		OK	
不正入力	g=5		OK	G に対して不適切
不正入力	alpha=1.01		OK	
不正入力	alpha=0.0		OK	
不正入力	alpha=”a”		OK	
不正入力	check_interval=0		OK	
不正入力	check_interval=”a”		OK	
不正入力	n_check_ok=0		OK	
不正入力	n_check_ok=”a”		OK	
不正入力	n_failure_max=0		OK	
不正入力	n_failure_max=”a”		OK	
不正入力	efficiency_log=0		OK	

※以下の設定をベースに、テスト項目に応じて設定を変更してテストを行いました。

interval = 3000

L2ckpt_freq = 20

L1ckpt_overhead = 100

L2ckpt_latency = 2000

ckptRestartTimes = [100,2000]

failRates = [1e-5,1e-6]

N = 1000　※G、g の設定範囲テスト時は 1000000

G = 4　※g の設定範囲テスト時は 1000000

g = 1

alpha = 1e-3

SN = 1000000

chk_interval = 1000

n_check_ok = 1

n_failure_max = 500000

efficiency_log = True

5.3 optimize_cr 関数

optimize_cr 関数の動作確認結果は以下の通りで、問題なく動作しております。

区分	テスト内容	結果確認内容	判定	備考
実行	最適化の動作確認 （デバッグ出力での処理確認）	最適化（デバッグ出力内容）が想定した処理、フローとなっていること	OK	デバッグ出力オン
設定範囲	L1ckpt_overhead=0	最適化が正常に終了すること	OK	
設定範囲	L1ckpt_overhead=100		OK	
設定範囲	L1ckpt_overhead=3600		OK	
設定範囲	L2ckpt_latency=0	最適化が正常に終了すること	OK	
設定範囲	L2ckpt_latency=1000		OK	
設定範囲	L2ckpt_latency=36000		OK	
設定範囲	ckptRestartTimes[0]=0	最適化が正常に終了すること	OK	
設定範囲	ckptRestartTimes[0]=100		OK	
設定範囲	ckptRestartTimes[0]=3600		OK	
設定範囲	ckptRestartTimes[1]=0	最適化が正常に終了すること	OK	
設定範囲	ckptRestartTimes[1]=1000		OK	
設定範囲	ckptRestartTimes[1]=36000		OK	
設定範囲	failRates[0]=0.0	最適化が正常に終了すること	OK	
設定範囲	failRates[0]=1.0×10^{-10}		OK	
設定範囲	failRates[0]=1.0×10^{-6}		OK	
設定範囲	failRates[1]=0.0	最適化が正常に終了すること	OK	
設定範囲	failRates[1]=1.0×10^{-10}		OK	
設定範囲	failRates[1]=1.0×10^{-6}		OK	
設定範囲	N=1	最適化が正常に終了すること	OK	
設定範囲	N=1000		OK	
設定範囲	N=1000000		OK	
設定範囲	G=1	最適化が正常に終了すること	OK	
設定範囲	G=1000		OK	
設定範囲	G=1000000		OK	
設定範囲	g=1	最適化が正常に終了すること	OK	
設定範囲	g=1000		OK	
設定範囲	g=1000000		OK	
設定範囲	alpha=1.0×10^{-2}	最適化が正常に終了すること	OK	
設定範囲	alpha=1.0×10^{-5}		OK	
設定範囲	alpha=1.0×10^{-7}		OK	
設定範囲	SN=10000	最適化が正常に終了すること	OK	

設定範囲	SN=1000000		OK	
設定範囲	n_steps=100	最適化が正常に終了すること	OK	
設定範囲	n_steps=5000		OK	
設定範囲	log_interval=0	最適化が正常に終了すること	OK	
設定範囲	log_interval=50		OK	
不正入力	L1ckpt_overhead=-1	エラーが発生すること	OK	
不正入力	L1ckpt_overhead=”a”	CUI に不正入力を示すメッセージが表示されること	OK	
不正入力	L2ckpt_latency=-1	返り値がすべて 0 になること	OK	
不正入力	L2ckpt_latency=”a”		OK	
不正入力	ckptRestartTimes=[-1,2000]		OK	
不正入力	ckptRestartTimes=[”a”,2000]		OK	
不正入力	ckptRestartTimes=[100,-1]		OK	
不正入力	ckptRestartTimes=[100,”a”]		OK	
不正入力	ckptRestartTimes=[100]		OK	
不正入力	ckptRestartTimes=100		OK	
不正入力	failRates=[-1,1e-6]		OK	
不正入力	failRates=[”a”,1e-6]		OK	
不正入力	failRates=[1e-5,-1]		OK	
不正入力	failRates=[1e-5,”a”]		OK	
不正入力	failRates=[1e-5]		OK	
不正入力	failRates=1e-5		OK	
不正入力	N=0		OK	
不正入力	N=”a”		OK	
不正入力	G=0		OK	
不正入力	G=”a”		OK	
不正入力	G=1001		OK	N に対して不適切
不正入力	g=0		OK	
不正入力	g=”a”		OK	
不正入力	g=5		OK	G に対して不適切
不正入力	alpha=1.01		OK	
不正入力	alpha=0.0		OK	
不正入力	alpha=”a”		OK	
不正入力	check_interval=0		OK	
不正入力	check_interval=”a”		OK	
不正入力	n_check_ok=0		OK	
不正入力	n_check_ok=”a”		OK	
不正入力	n_failure_max=0		OK	
不正入力	n_failure_max=”a”		OK	

不正入力	n_steps=0		OK	
不正入力	n_steps=”a”		OK	
不正入力	log_interval=-1		OK	
不正入力	log_interval=”a”		OK	

※以下の設定をベースに、テスト項目に応じて設定を変更してテストを行いました。

L1ckpt_overhead = 100

L2ckpt_latency = 2000

ckptRestartTimes = [100,2000]

failRates = [1e-5,1e-6]

N = 1000　※G、g の設定範囲テスト時は 1000000

G = 4　※g の設定範囲テスト時は 1000000

g = 1

SN = 1000000

alpha = 1e-3

chk_interval = 1000

n_check_ok = 1

n_steps = 250

log_interval = 50

5.4　Efficiency 変化グラフ

以下に simulate_cr 関数で interval と L2_ckpt_freq を変更した際の Efficiency 変化のグラフを示します。

L1ckpt_overhead = 10　L2ckpt_latency = 100

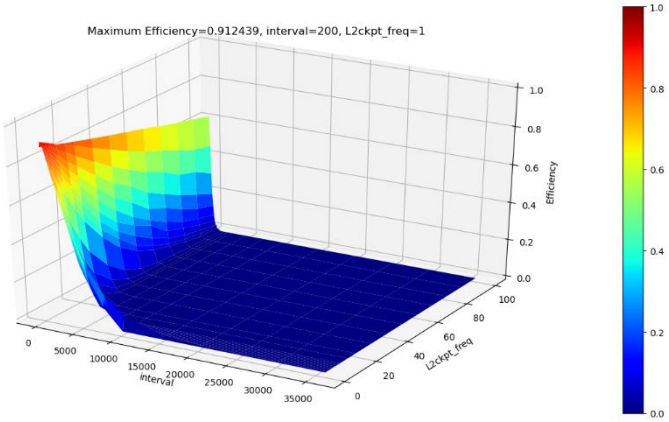
ckptRestartTimes = [10,100]

N = 1000　G = 4　g = 1　SN = 1000000

alpha = 1e-4　chk_interval = 1000　n_check_ok = 1

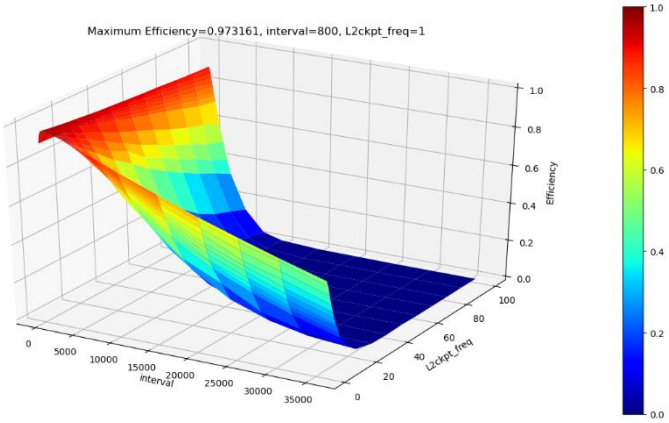
failRates = [1e-4,1e-5]

optimize_cr の最適化結果:
　efficiency = 0.912773
　interval = 250
　L2ckpt_freq = 1

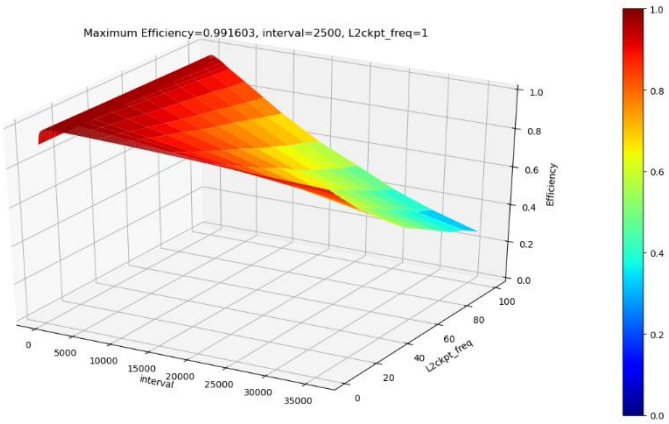


failRates = [1e-5,1e-6]

optimize_cr の最適化結果:
　efficiency = 0.973213
　interval = 680
　L2ckpt_freq = 1



failRates = [1e-6,1e-7]



L1ckpt_overhead = 10　L2ckpt_latency = 100

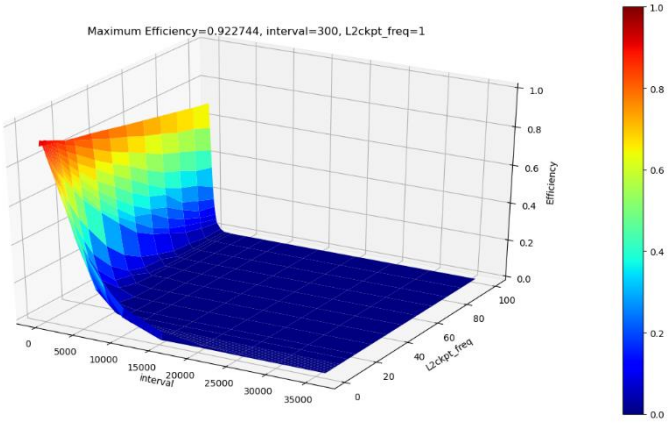
ckptRestartTimes = [10,100]

N = 1000　G = 4　g = 2　SN = 1000000

alpha = 1e-4　chk_interval = 1000　n_check_ok = 1

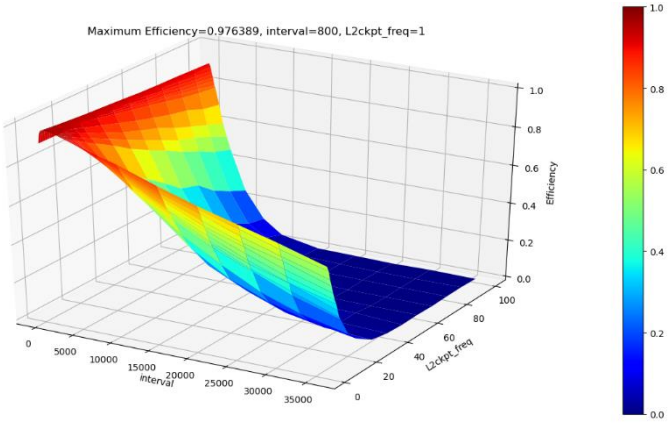
failRates = [1e-4,1e-5]

optimize_cr の最適化結果:
　efficiency = 0.922973
　interval = 273
　L2ckpt_freq = 1

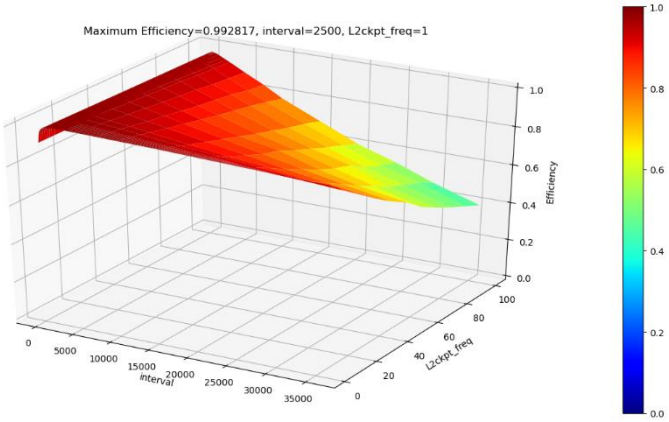


failRates = [1e-5,1e-6]

optimize_cr の最適化結果:
　efficiency = 0.976001
　interval = 735
　L2ckpt_freq = 1



failRates = [1e-6,1e-7]



L1ckpt_overhead = 10　L2ckpt_latency = 100

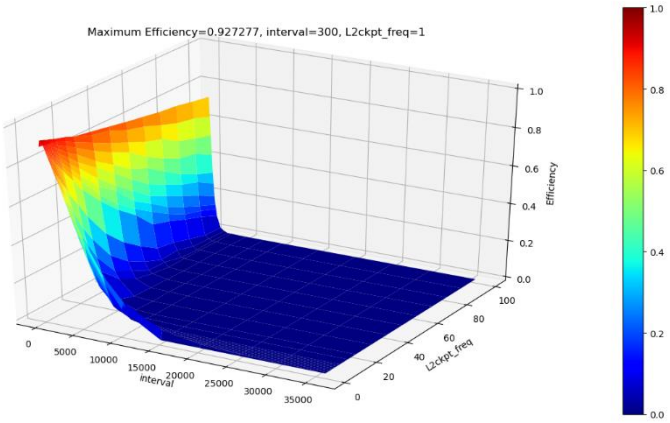
ckptRestartTimes = [10,100]

N = 1000　G = 4　g = 3　SN = 1000000

alpha = 1e-4　chk_interval = 1000　n_check_ok = 1

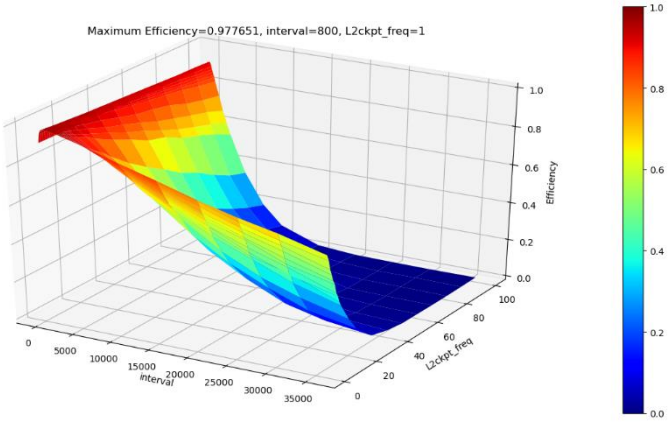
failRates = [1e-4,1e-5]

optimize_cr の最適化結果:
　efficiency = 0.926871
　interval = 310
　L2ckpt_freq = 1

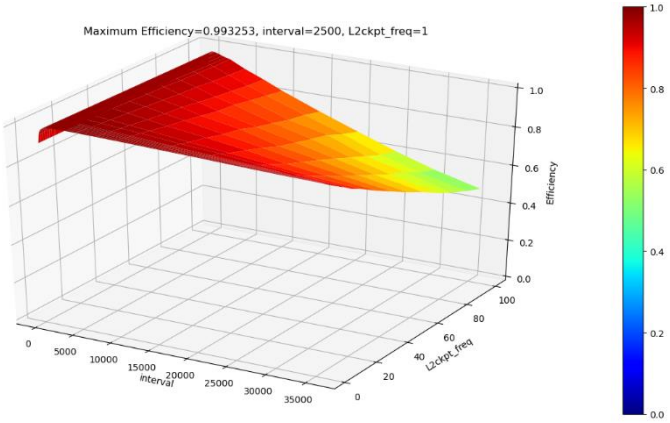


failRates = [1e-5,1e-6]

optimize_cr の最適化結果:
　efficiency = 0.978218
　interval = 963
　L2ckpt_freq = 1



failRates = [1e-6,1e-7]



L1ckpt_overhead = 10　L2ckpt_latency = 100

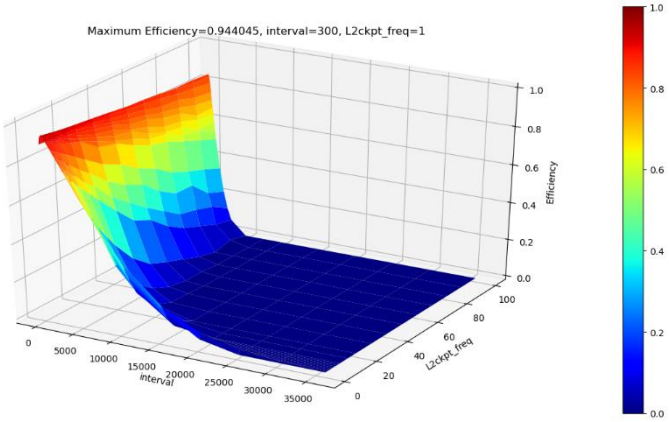
ckptRestartTimes = [10,100]

N = 1000　G = 1　g = 1　SN = 1000000

alpha = 1e-4　chk_interval = 1000　n_check_ok = 1

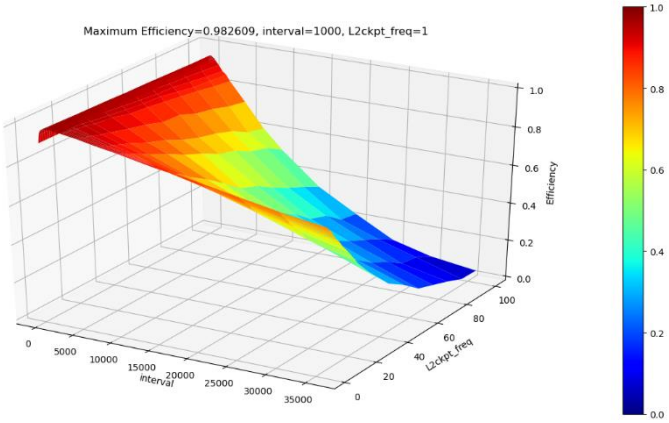
failRates = [1e-4,1e-5]

optimize_cr の最適化結果:
　efficiency = 0.944331
　interval = 329
　L2ckpt_freq = 1

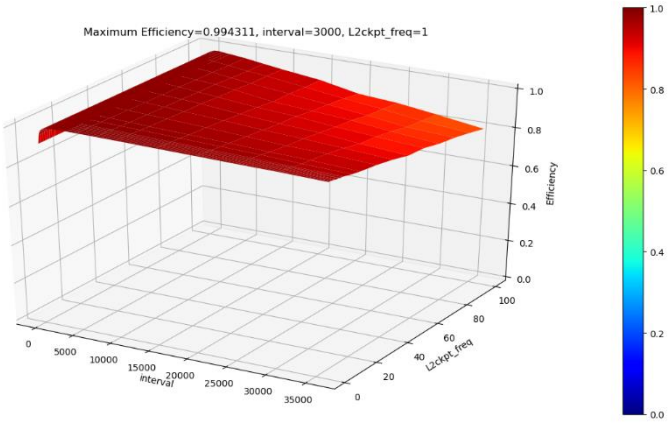


failRates = [1e-5,1e-6]

optimize_cr の最適化結果:
　efficiency = 0.982835
　interval = 1184
　L2ckpt_freq = 1



failRates = [1e-6,1e-7]



L1ckpt_overhead = 200　L2ckpt_latency = 6000

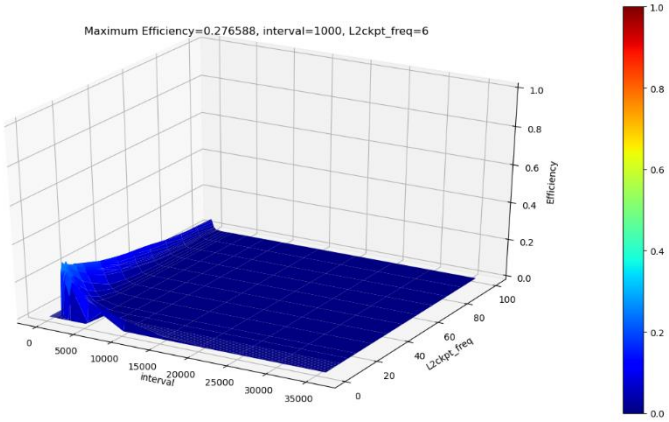
ckptRestartTimes = [200,6000]

N = 1000　G = 4　g = 1　SN = 1000000

alpha = 1e-4　chk_interval = 1000　n_check_ok = 1

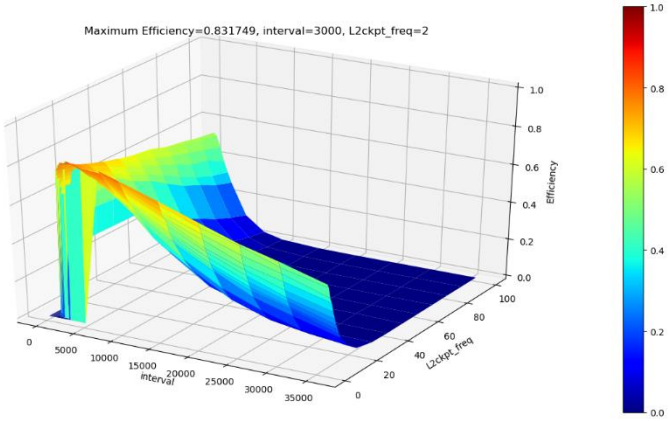
failRates = [1e-4,1e-5]

optimize_cr の最適化結果:
　efficiency = 0.261560
　interval = 1309
　L2ckpt_freq = 5

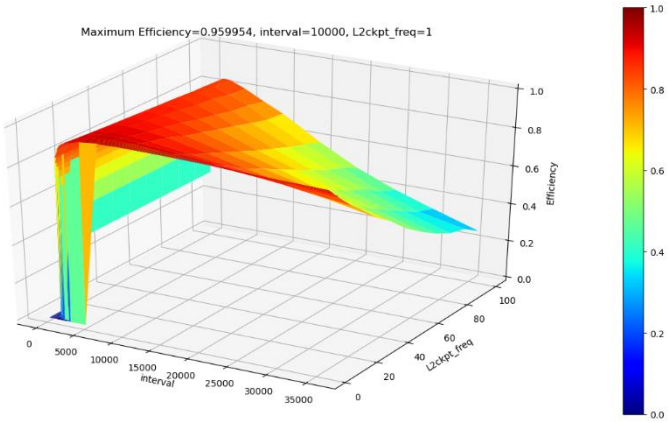


failRates = [1e-5,1e-6]

optimize_cr の最適化結果:
　efficiency = 0.839032
　interval = 5837
　L2ckpt_freq = 1



failRates = [1e-6,1e-7]



L1ckpt_overhead = 100　L2ckpt_latency = 2000

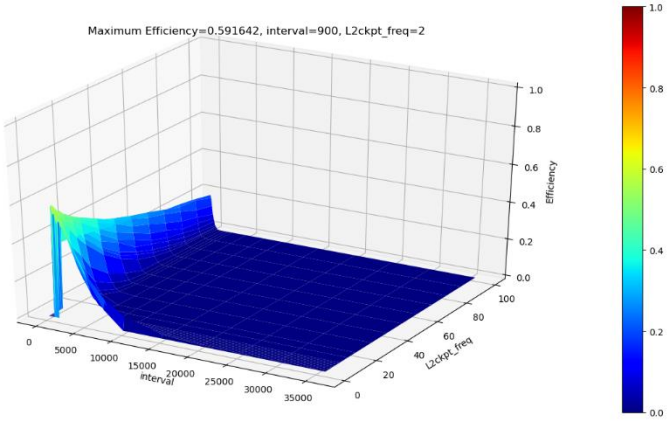
ckptRestartTimes = [100,2000]

N = 1000　G = 4　g = 1　SN = 1000000

alpha = 1e-4　chk_interval = 1000　n_check_ok = 1

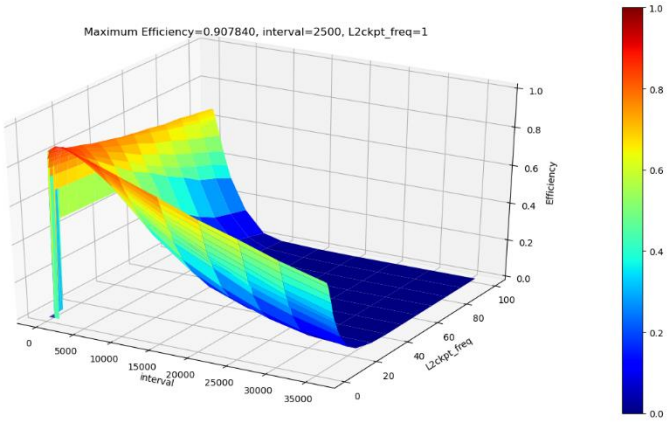
failRates = [1e-4,1e-5]

optimize_cr の最適化結果:
　efficiency = 0.594987
　interval = 907
　L2ckpt_freq = 2

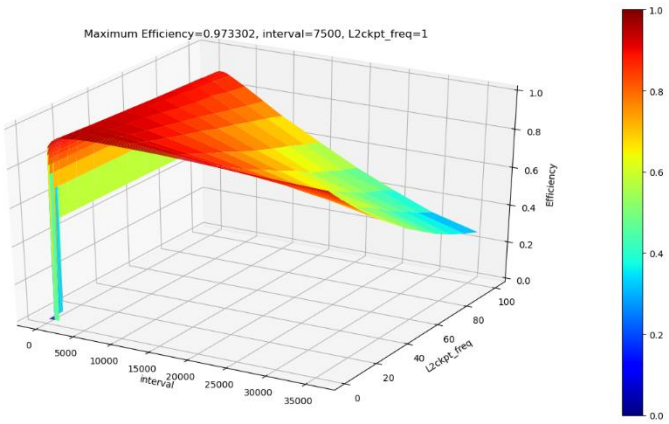


failRates = [1e-5,1e-6]

optimize_cr の最適化結果:
　efficiency = 0.908087
　interval = 2281
　L2ckpt_freq = 1



failRates = [1e-6,1e-7]



6 関数使用例

以下に simulate_cr 関数、optimize_cr 関数の使用例を示します。

```
from checkpoint_restart_simulator import *

def main():

    mode = 0

    interval = 2000
    L2ckpt_freq = 4
    L1ckpt_overhead = 200
    L2ckpt_latency = 6000
    ckptRestartTimes = [200, 6000]
    failRates = [1e-05, 1e-06]
    N = 1000
    SN = 1000000
    G = 4
    g = 1
    alpha = 1e-4
    chk_interval = 1000
    n_check_ok = 1

    # simulation
    if mode == 0:
        print("*** Simulation Start ***")
        rtn_vals = simulate_cr(interval, L2ckpt_freq, L1ckpt_overhead,
            L2ckpt_latency, ckptRestartTimes, failRates, N, SN, G, g, alpha,
            check_interval=chk_interval, n_check_ok=n_check_ok,
            efficiency_log=True)

        print("efficiency               = %f" % rtn_vals[0])
        print("actual computation time  = %f" % rtn_vals[1])
        print("total compute state time = %f" % rtn_vals[2])
        print("total L1 overhead time   = %f" % rtn_vals[3])
        print("total L1 recovery time   = %f" % rtn_vals[4])
        print("total L2 overhead time   = %f" % rtn_vals[5])
        print("total L2 recovery time   = %f" % rtn_vals[6])
        print("*** Simulation Complete ***")

    # optimization
    else:
        print("*** Optimization Start ***")
        rtn_vals = optimize_cr(L1ckpt_overhead, L2ckpt_latency,
            ckptRestartTimes, failRates, N, SN, G, g, alpha,
            check_interval=chk_interval, n_check_ok=n_check_ok,
            n_steps=3000, log_interval=100)

        print("¥ninterval = %d  L2ckpt_freq = %d" % (rtn_vals[7], rtn_vals[8]))
        print("efficiency               = %f" % rtn_vals[0])
        print("actual computation time  = %f" % rtn_vals[1])
        print("total compute state time = %f" % rtn_vals[2])
        print("total L1 overhead time   = %f" % rtn_vals[3])
        print("total L1 recovery time   = %f" % rtn_vals[4])
        print("total L2 overhead time   = %f" % rtn_vals[5])
        print("total L2 recovery time   = %f" % rtn_vals[6])
        print("*** Optimization Complete ***")

if __name__  == "__main__":
    main()
```