

# As Much Resilience As You Want: A Resilient Legion

Karthik Murthy, Mike Bauer, Todd Warszawski, Wonchan Lee, Elliott  
Slaughter, Sean Treichler, and Alex Aiken

Stanford University, USA,

**Abstract.** Resilient is an important aspect of scaling applications on large clusters. As we approach parallelism profiles of several millions and long running applications, we need to ensure that ineffect “we reach the end”. Defining the policy interaction with the programming model features necessitates that we revisit the memory consistency model. Recoverability, a critical step in resilience, opens the door to optimizations such as speculation. We also evaluate this. . . .

**Keywords:** resilience

## 1 Introduction

## 2 Resilience Policy

### 3 Interaction of Resilience Policy with Legion Features

which are the legion features of importance ?

put a figure of interaction.

what is the memory consistency angle here ?

what does it mean to advance the commit wavefront

**Resilience is a tangling of the lifetime of a task and a region snapshot**

1) when can we advance a commit wavefront ? 2) whats the lifetime of a region-instance snapshot ?

on this front, we see it as a two-step process a) define a consistent cut of tasks problem, b) commit any task strictly behind this wavefront c) garbage collect any snapshot that serves as input to any task that is already committed

consistent cut of tasks that can be part of the commit wavefront: a set of tasks whose inputs are need\_preserve'ed, or they are strictly post-dominated by tasks whose inputs are need\_preserved.

3) what about copy/index/tasks ? copy local to local follows the above semantics copy local to remote get committed immediately after successful execution. index launch tasks, actually feel need not be in the task graph, unless virtual mapping is used. they can be garbage collected immediately after all child tasks are included in the dependence analysis wavefront - I am thinking we will never have a case where are relauching the index launch, since hte child tasks either are part of the committed wavefront or are not (pending a discussion on phase barriers)

**What to do about must\_epoch tasks with phase\_barrier inside them ?**

answer: restartable phase\_barrier with generation commit callback

## 4 Resilience Application: Speculation

there are three different wavefronts in Legion, we can speculate on any of them.

From a speculation perspective, are they different ? Are we novel, since we have these three different wavefronts ? Can we navigate through this, like the blanks

1) execution wavefront what does it mean to speculate here, does the other steps have to be complete before we do this.

2) mapping wavefront

3) dependence analysis wavefront

– see mike's 6 wavefront answer.

There are three different wavefronts, there could

**the 6 wavefronts, where does speculation plays a role** mike - speculation is more about tracking the resolution wavefront while resilience is more about tracking the commit wavefront, but the when things go bad, then i think the machinery to restart the mapping and execution wavefronts should be the same

## 5 Implementation

## 6 Experiments

### 6.1 Index Tasks

*Growth of memory as execution proceeds*

*Performance overhead as execution proceeds*

Number of Iterations (1000 tasks/index launch, seconds)	Time Without Resilience	Without lg:resilient	Time
100	7.2		
200	13.8		
400	26.1		
800	55		
1000	65		

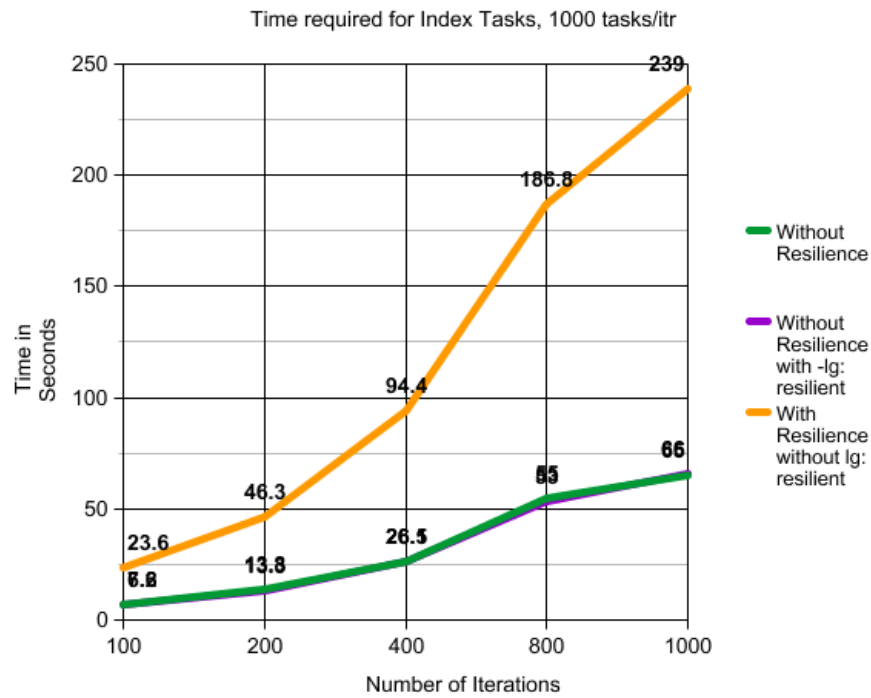
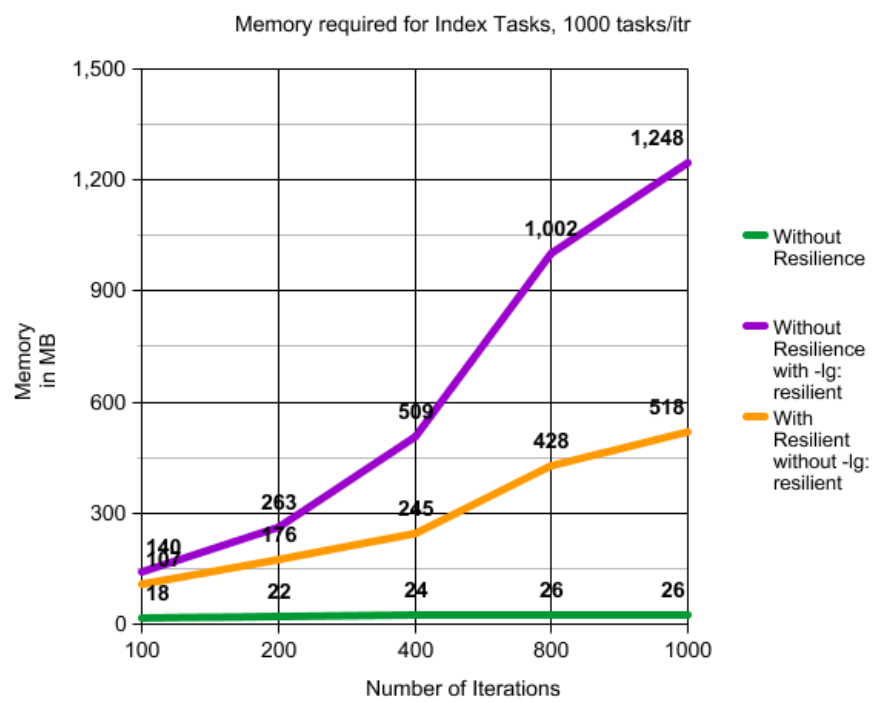


Fig. 1. Total time taken by 02\_index\_tasks.



**Fig. 2.** Total Memory footprint by 02.index\_tasks.

Number of Iterations (1000 tasks/index launch, MB)	Memory% Without Resilience	Without lg:resilient Mem
100	18	
200	22	
400	24	
800	26	
1000	26	

## 6.2 Stencil

## 6.3 local recover vs global recovery

## 6.4 compute/comm vs no-failure/single failure/multi-failure

## 6.5 S3D, Pennant, Stencil, Circuit

## 6.6 Some Interesting Task Graphs for Recovery

## **6.7 Experiments**

## **7 Adaptive Resilience**

### **7.1 Resilience Policy Implemented via Mapper: Example 1 Generic**

### **7.2 Resilience Policy Implemented via Mapper: Example 2 UT Austin**

Interaction of `need_preserve` with the commit wavefront

Obtaining dependence graph in the mapper before calling `need_preserve`

## **8 Related Work**

## **9 Conclusion**

## **References**

1. Clarke, F., Ekeland, I.: Nonlinear oscillations and boundary-value problems for Hamiltonian systems. Arch. Rat. Mech. Anal. 78, 315–333 (1982)
2. Rabinowitz, P.: On subharmonic solutions of a Hamiltonian system. Comm. Pure Appl. Math. 33, 609–633 (1980)