# Phase E Scalability Analysis

## Performance Debrief & Data Review

January 2026

Status: ✅ Analysis Complete
Source: Perf__Phase_E__Scalability test suite

# Executive Summary: System Health & Key Findings

### Scaling Behaviour

**O(n) Linear**

Verdict

Algorithm is sound; no $O(n^2)$ issues detected.

### Optimization Win

**1.9x–2.2x**

Speedup

fast_create mode delivers ~50% overall improvement consistently.

### Primary Bottleneck

**Dict→MGraph**

78–95% of Time

The critical path for future optimization.

Note: Fixed overhead identified at ~9ms minimum (Architectural).

# Methodology & Test Scope

Analysis conducted using 87-iteration Fibonacci sampling (measure_fast) to ensure statistical significance.

## Quick Analysis

Scope: 10, 50, 100 nodes

Result: **49.8%** improvement detected

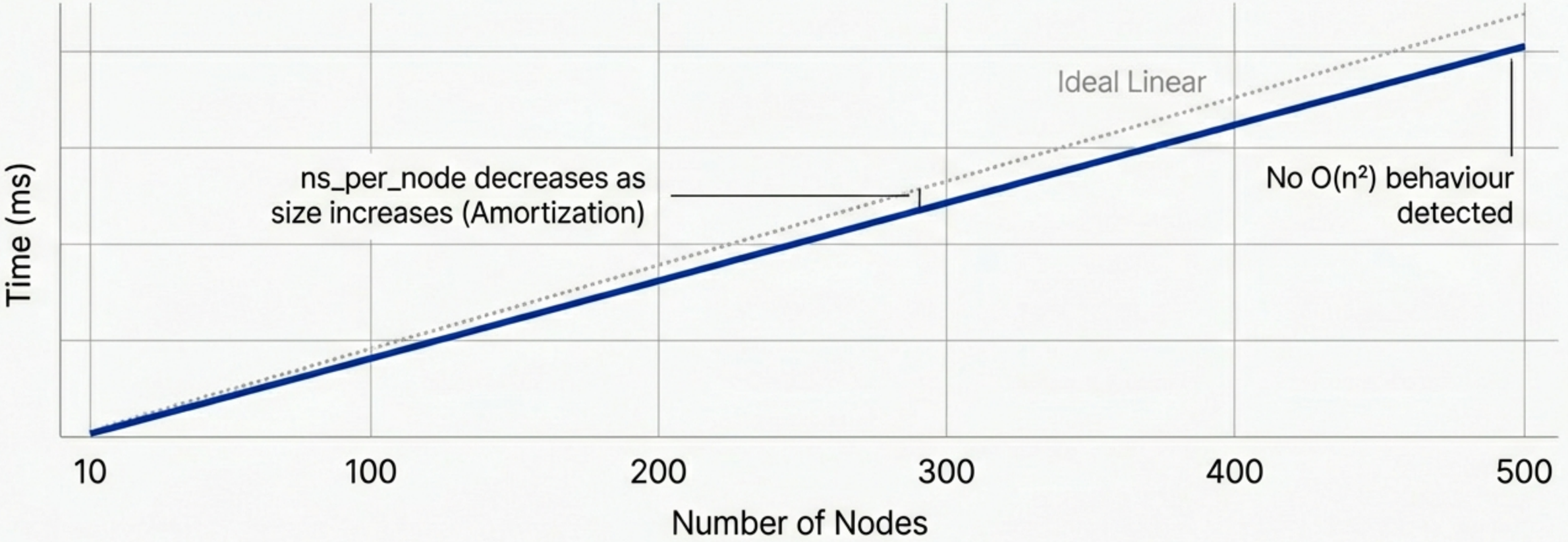
## Standard Analysis

Scope: 10, 50, 100, 500 nodes

Result: **49.2%** improvement detected

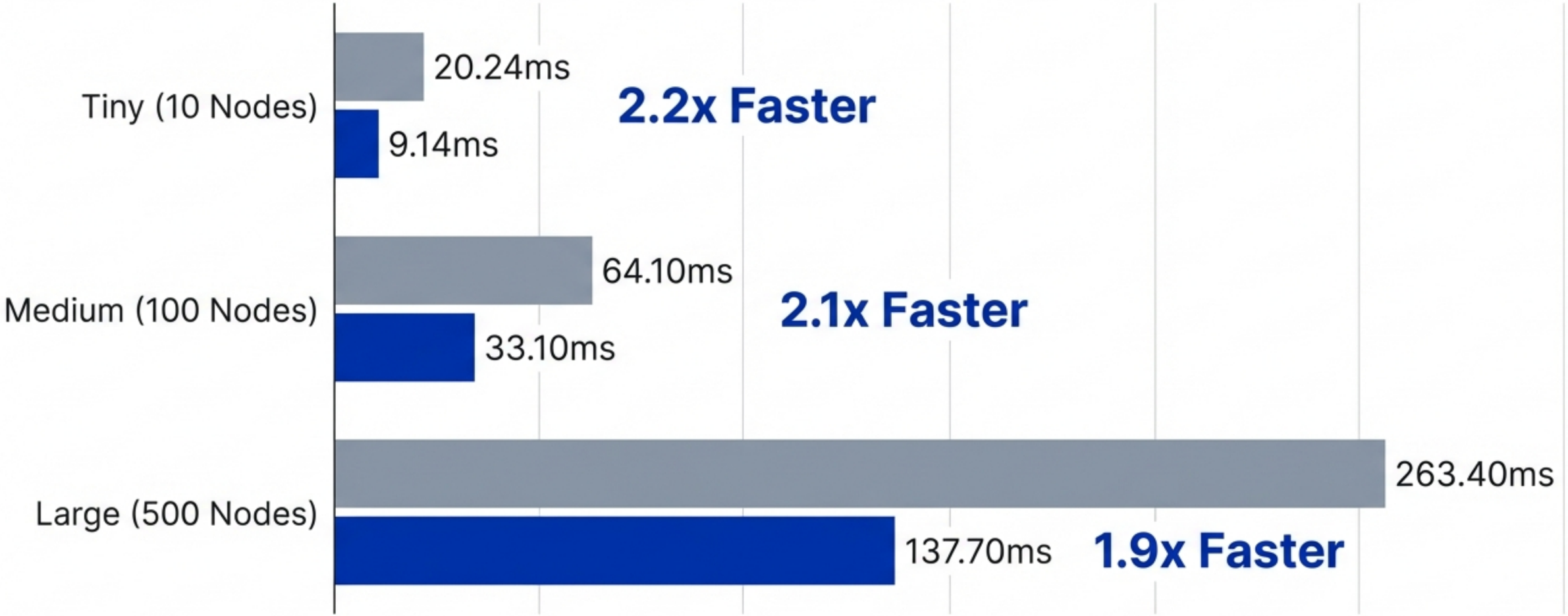
Insight: Both analysis modes produce consistent verdicts. ‘Quick’ analysis is sufficient for general profiling; ‘Standard’ is required only when investigating large-scale behaviour.

# Core Finding: Healthy O(n) Scalability




The algorithm performs better than pure linear scaling due to amortization of fixed overheads. The system becomes more efficient per-node as the document size grows.

# The Win: Validation of fast_create




The optimization consistently halves execution time across all tested scales.

# Speedup Consistency Across Scales

| Size | Nodes | Default Time | Fast Create Time | Speedup |
|---|---|---|---|---|
| Tiny | 10 | 20.94ms | 9.44ms | **2.2x** |
| Small | 50 | 40.40ms | 19.60ms | **2.1x** |
| Medium | 100 | 71.00ms | 33.30ms | **2.1x** |
| Large | 500 | 266.60ms | 140.30ms | **1.9x** |

Analysis: The slight decrease at 500 nodes suggests irreducible complexity in per-node conversion logic, but the gain remains substantial (~50% time reduction).

# The Bottleneck is Dict→MGraph

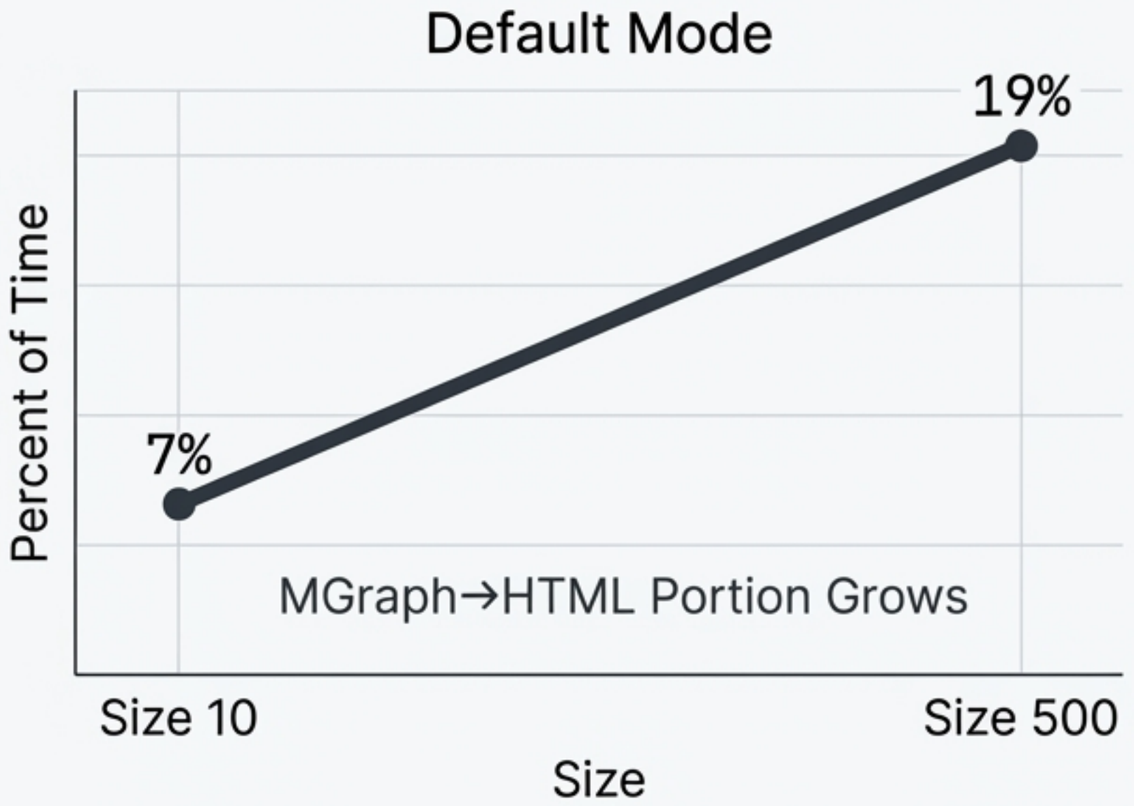

Tiny Scale (10 nodes):
92% of total time

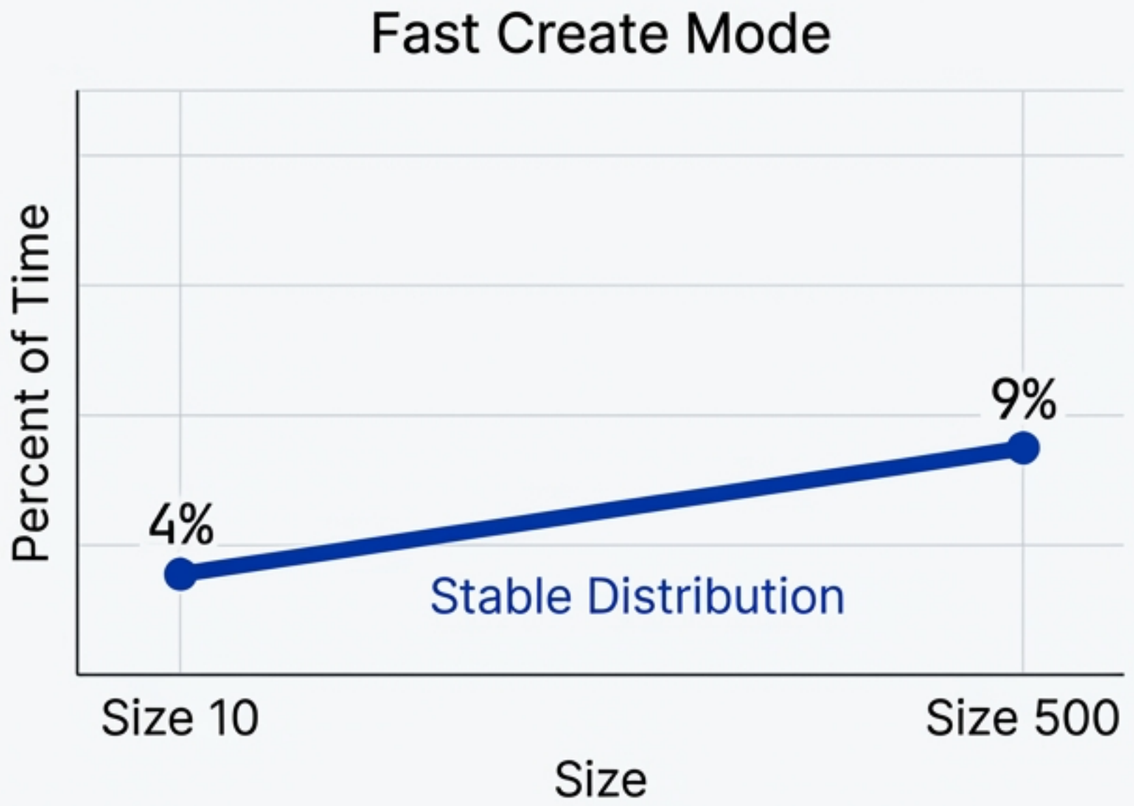
Large Scale (500 nodes):
81% of total time

“This is the only stage worth optimizing further.”

# Stage Distribution Shift & Stability

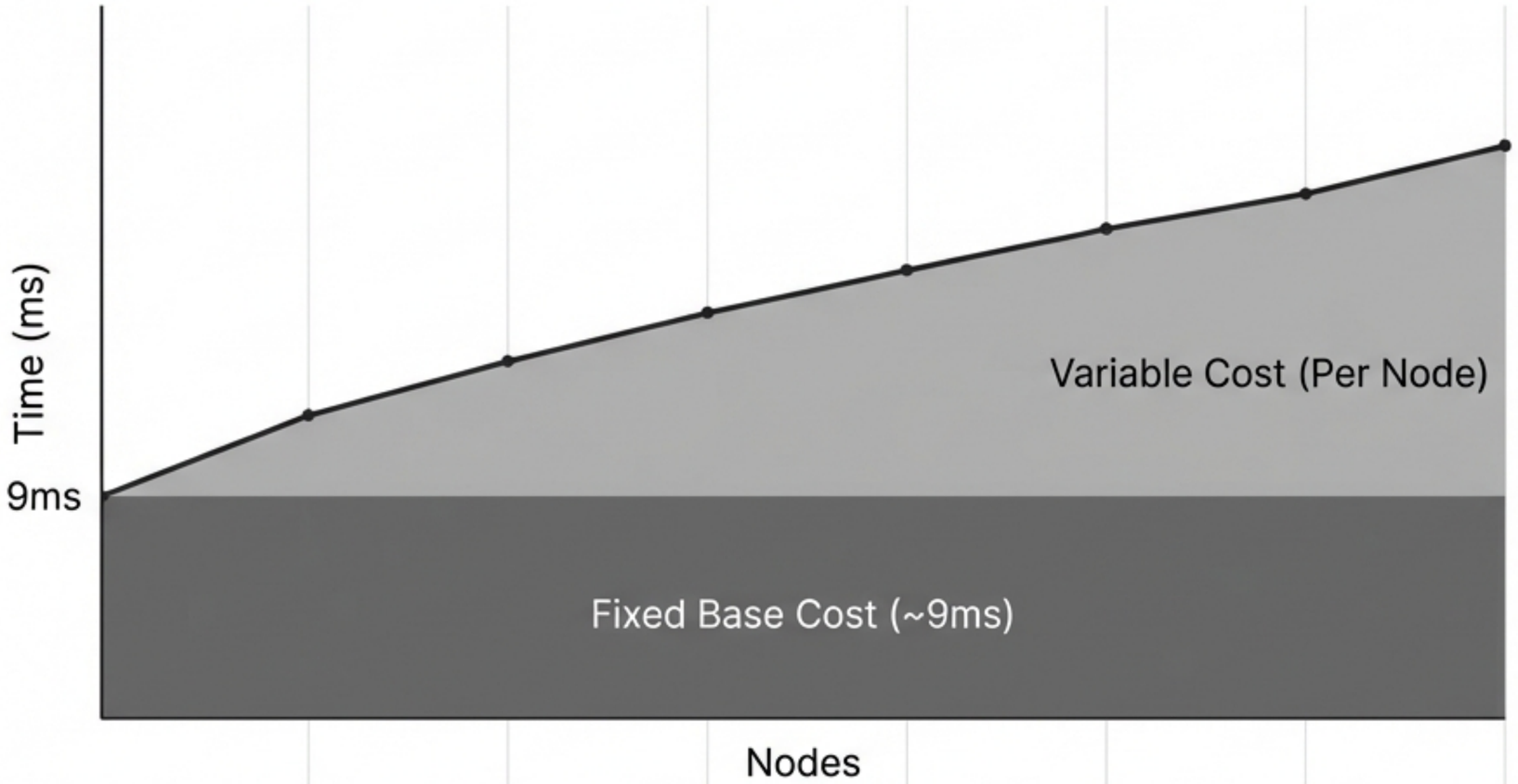
## Comparing behaviour of MGraph→HTML stage




fast_create benefits the HTML reconstruction phase more at larger sizes, likely because it handles Type_Safe object creation more efficiently.

# The ‘Fixed Overhead’ Floor




Minimum floor: **~9ms**
(even 1-node HTML takes 7ms)

Marginal Cost (10 nodes):
**~900µs** per node

Marginal Cost (500 nodes):
**~275µs** per node

The high initial cost is driven by document structure creation, index initialization, and MGraph infrastructure setup.

# Per-Node Efficiency (Amortization)

**2,024,000 ns**

Per Node Cost (Tiny Scale)

**526,800 ns**

Per Node Cost (Large Scale)

As the node count increases, the fixed overhead is spread out, resulting in a 74% reduction in per-node cost at scale. This confirms the architecture scales scales efficiently.

# Recommendation: Production Deployment

## Always enable fast_create.

✅ Consistent 1.9x–2.2x speedup.

✅ No behavioral changes or regressions.

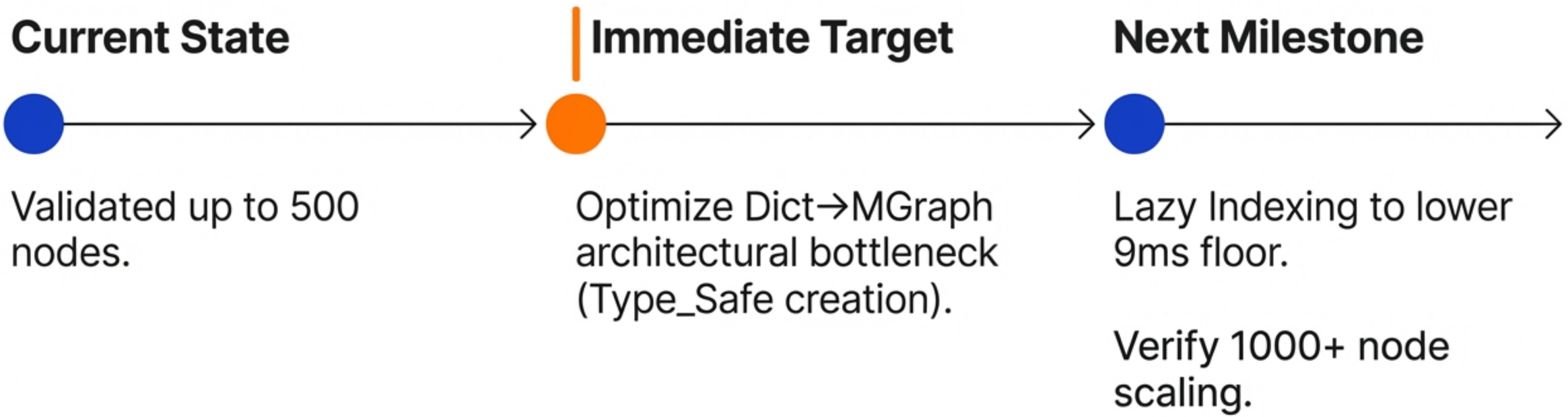
✅ Improves stability of MGraph→HTML at scale.

Result: Immediate ~50% performance gain for all users.

# Engineering Roadmap & Priorities

| Priority | Task | Goal |
| --- | --- | --- |
| P0 (Critical) | Profile inside `convert_from_dict()` | Identify specific object creation costs. |
| P1 | Count MGraph objects created per node | Understand the multiplication factor. |
| P2 | Test lazy index initialization | Reduce the ~9ms fixed overhead. |
| P3 | Benchmark at 1000+ nodes | Verify scaling holds at extra-large sizes. |

# Future Outlook



# Generated Report Artifacts

- `scaling__quick__default.txt`
- `scaling__quick__fast_create.txt`
- `scaling__standard__default.txt`
- `scaling__standard__fast_create.txt`
- `scaling__comparison__quick.txt`
- `scaling__comparison__standard.txt`

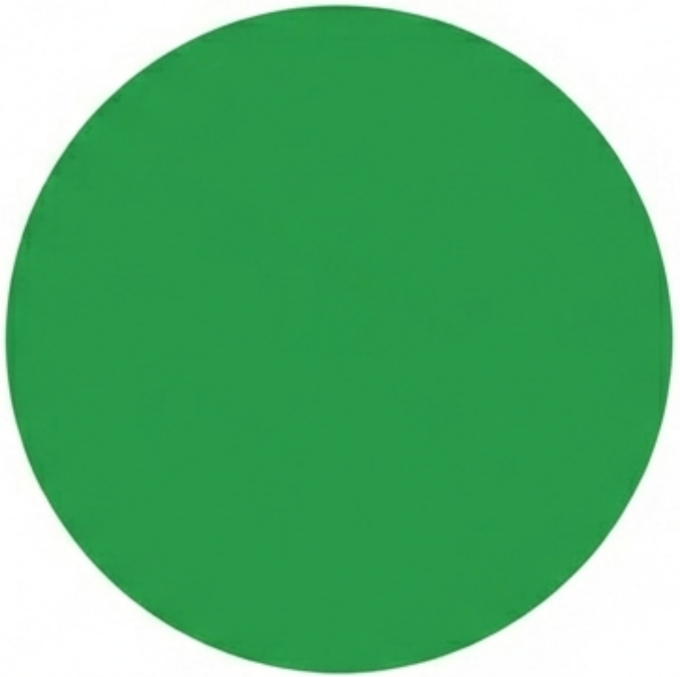
All measurements used the **Perf__Phase_E__Scalability** test suite.

# Final Verdict

**Algorithm is Sound:** O(n) linear scaling verified.

**Speedup is Real:** fast_create delivers a consistent 2x improvement.

**Target is Defined:** Future gains lie exclusively in optimizing Dict→MGraph object creation.



Phase E Analysis Complete.