SOP: Token Optimization for Marketing Analysis System (2025)

*Generated: 03/09/2025 14:16*

# SOP: Token Optimization for Marketing Analysis System (2025)

: 1.0

: 03/09/2025

: Implement 2025 token optimization best practices to achieve 40-70% cost savings

: Technical Research Analysis Report - 2025 Token Optimization Best Practices

---

## \*\*OVERVIEW & OBJECTIVES\*\*

### \*\*Target Improvements:\*\*

* \*\*40-70% token cost reduction\*\* through strategic optimization
* \*\*50% reduction in processing time\*\* via parallel execution
* \*\*90% efficiency improvement\*\* for repetitive queries via caching
* \*\*60% reduction in system overhead\*\* through MCP optimization

### \*\*Implementation Priority:\*\*

1. \*\*Immediate (Week 1)\*\*: Prompt optimization & parallel tool execution
2. \*\*Short-term (Week 2-3)\*\*: Context caching & batching strategies
3. \*\*Medium-term (Week 4-6)\*\*: Agent orchestration & workflow optimization

---

## \*\*SECTION 1: PROMPT ENGINEERING OPTIMIZATION\*\*

### \*\*1.1 40% Token Compression Challenge\*\*

: Every system prompt must be compressed by 40% while maintaining performance

:

Please conduct a comprehensive analysis of this website including technical SEO, user experience, performance metrics, accessibility compliance, content quality, and competitive positioning. I need you to examine all aspects thoroughly and provide detailed recommendations.

:

# Task  
Analyze website: technical SEO + UX + performance + accessibility + content + competitive positioning  
  
## Output  
- Technical findings with metrics  
- UX issues with evidence   
- Performance scores  
- Actionable recommendations

### \*\*1.2 Structured Tag Implementation\*\*

<task>  
[Specific action required]  
</task>  
  
<context>  
[Essential background only]  
</context>  
  
<output\_format>  
[Exact structure required]  
</output\_format>  
  
<success\_criteria>  
[Measurable outcomes]  
</success\_criteria>

### \*\*1.3 Few-Shot Learning Efficiency\*\*

: Include 1-3 examples maximum for complex tasks

:

<examples>  
Input: [Website URL]  
Output: [Expected format with 1 concrete example]  
</examples>

---

## \*\*SECTION 2: MCP TOOL OPTIMIZATION\*\*

### \*\*2.1 Parallel Tool Execution Strategy\*\*

: Batch independent tool calls in single messages for 50% time reduction

:

# Single message with multiple tool calls  
def optimize\_parallel\_execution():  
 # Batch related operations  
 tools = [  
 ("WebFetch", website\_url),  
 ("WebSearch", "competitor analysis query"),  
 ("Read", "existing\_report.md")  
 ]  
 # Execute in parallel within single agent call  
 return batch\_execute\_tools(tools)

:

* [ ] Group related tool calls together
* [ ] Execute independent analyses simultaneously
* [ ] Use single message for multiple tool invocations
* [ ] Combine file reads with web requests

### \*\*2.2 Context-Aware Tool Selection\*\*

(Use most efficient tool for each task):

|  |  |  |  |
| --- | --- | --- | --- |
| **Task Type** | **Primary Tool** | **Token Cost** | **Efficiency Score** |
| Website content extraction | WebFetch | Low | 95% |
| Multi-page crawling | Scrapy (custom) | Medium | 90% |
| Performance testing | Playwright MCP | High | 85% |
| File operations | Read/Write batch | Very Low | 98% |
| Search operations | WebSearch | Medium | 90% |

:

1. Always use lowest-cost tool that meets requirements
2. Batch multiple file operations
3. Combine search queries when possible
4. Cache frequently used results

### \*\*2.3 Result Caching & Memoization\*\*

:

# Cache expensive operations  
CACHE\_DURATION = {  
 "website\_crawl": "24\_hours",  
 "competitor\_analysis": "7\_days",   
 "keyword\_research": "3\_days",  
 "performance\_data": "1\_hour"  
}  
  
def cached\_analysis(url, analysis\_type):  
 cache\_key = f"{url}\_{analysis\_type}\_{date}"  
 if cache\_exists(cache\_key):  
 return load\_cache(cache\_key) # 90% token savings  
 else:  
 result = run\_analysis(url, analysis\_type)  
 save\_cache(cache\_key, result)  
 return result

---

## \*\*SECTION 3: AGENT ORCHESTRATION EFFICIENCY\*\*

### \*\*3.1 Agent Selection Optimization\*\*

:

|  |  |  |  |
| --- | --- | --- | --- |
| **Analysis Type** | **Primary Agent** | **Fallback Agent** | **Token Savings** |
| SEO Analysis | `technical\_seo\_analyst` | `seo\_strategist` | 30% |
| UX Testing | `ux-ui-analyst` | `ux\_flow\_validator` | 25% |
| Content Creation | `content\_generator` | `content\_strategist` | 35% |
| Performance Analysis | `performance\_tester` | `technical\_seo\_analyst` | 40% |

:

1. Always use primary agent for best token efficiency
2. Only invoke fallback agents if primary fails
3. Document agent selection rationale
4. Monitor performance metrics for optimization

### \*\*3.2 Context Sharing Between Agents\*\*

:

# Context Template (Use for all agent handoffs)  
  
## Previous Analysis Summary  
- Key findings: [3 bullet points maximum]  
- Data collected: [File references only]  
- Next required actions: [Specific tasks]  
  
## Shared Resources   
- Files: [List paths, not content]  
- URLs: [Reference list]  
- Tools used: [Name only]  
  
## Success Criteria  
- [Measurable outcomes expected]

:

* [ ] Pass file references, not full content
* [ ] Summarize findings in 3 points maximum
* [ ] Share tool outputs via file paths
* [ ] Use consistent context templates

### \*\*3.3 Sequential vs Parallel Execution Strategy\*\*

:

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario** | **Strategy** | **Token Savings** | **Implementation** |
| \*\*Independent analyses\*\* | Parallel | 50% | Single message, multiple tool calls |
| \*\*Dependent workflows\*\* | Sequential | 20% | Pass minimal context between stages |
| \*\*Data collection\*\* | Parallel | 60% | Batch all web requests together |
| \*\*Report generation\*\* | Sequential | 15% | Build on previous outputs |

:

1. \*\*Phase 1 (Parallel)\*\*: Data collection - WebFetch, Scrapy, WebSearch simultaneously
2. \*\*Phase 2 (Parallel)\*\*: Analysis - Technical, UX, Content agents simultaneously
3. \*\*Phase 3 (Sequential)\*\*: Report generation - Build on analysis results
4. \*\*Phase 4 (Parallel)\*\*: Export & Distribution - Multiple format generation

---

## \*\*SECTION 4: DOCUMENTATION & REPORTING EFFICIENCY\*\*

### \*\*4.1 Template-Based Content Creation\*\*

(70% time savings):

# Analysis Report Template  
\*\*Website\*\*: {{WEBSITE\_URL}}  
\*\*Date\*\*: {{ANALYSIS\_DATE}}  
\*\*Analyst\*\*: {{AGENT\_NAME}}  
  
## Key Findings  
{{TOP\_3\_FINDINGS}}  
  
## Technical Metrics   
{{PERFORMANCE\_TABLE}}  
  
## Recommendations  
{{ACTION\_ITEMS}}  
  
---  
\*Generated by: {{SYSTEM\_NAME}} | Confidence: {{CONFIDENCE\_LEVEL}}\*

:

* Use templates for all standard reports
* Populate variables programmatically
* Maintain consistent formatting
* Include generation metadata

### \*\*4.2 Token-Efficient Markup Strategies\*\*

:

# Use Tables for Data (Not Lists)  
| Metric | Value | Status |  
|--------|-------|--------|  
| Load Time | 2.1s | ✅ Good |  
  
# Use Abbreviations  
- SEO (not Search Engine Optimization)  
- UX (not User Experience)   
- CTA (not Call-to-Action)  
  
# Compress Repeated Phrases  
- Results: 28 pages analyzed (not "The analysis included 28 pages")  
- Score: 85/100 (not "The score achieved was 85 out of 100")

### \*\*4.3 Automated Documentation Workflows\*\*

:

def generate\_efficient\_report(data):  
 template = load\_template("standard\_analysis.md")  
 variables = extract\_key\_metrics(data)  
 report = populate\_template(template, variables)  
 return compress\_content(report, target\_reduction=0.4)

:

* [ ] Use templates for all standard outputs
* [ ] Compress content by 40% without losing meaning
* [ ] Include only essential details
* [ ] Generate metadata automatically

---

## \*\*SECTION 5: CRAWLING & DATA EXTRACTION OPTIMIZATION\*\*

### \*\*5.1 Efficient Web Crawling Patterns\*\*

:

# High-efficiency settings  
SCRAPY\_SETTINGS = {  
 'CONCURRENT\_REQUESTS': 16, # Parallel requests  
 'DOWNLOAD\_DELAY': 0.5, # Balance speed vs politeness   
 'AUTOTHROTTLE\_ENABLED': True, # Adaptive throttling  
 'AUTOTHROTTLE\_TARGET\_CONCURRENCY': 2.0,  
 'COMPRESSION\_ENABLED': True, # Reduce bandwidth  
 'HTTPCACHE\_ENABLED': True, # Cache responses  
}

:

1. Extract only required fields (not entire pages)
2. Use XPath selectors for efficiency
3. Implement data validation during extraction
4. Cache extracted data for reuse

### \*\*5.2 Batch Processing Techniques\*\*

:

def batch\_url\_analysis(urls):  
 # Group URLs by domain for efficiency  
 grouped\_urls = group\_by\_domain(urls)  
   
 # Process in batches of 10  
 batch\_size = 10  
 results = []  
   
 for batch in chunk\_list(grouped\_urls, batch\_size):  
 batch\_results = parallel\_process(batch)  
 results.extend(batch\_results)  
   
 return consolidate\_results(results)

### \*\*5.3 Memory-Efficient Data Handling\*\*

:

* Stream large datasets instead of loading entirely in memory
* Use generators for data processing pipelines
* Clean up temporary files immediately after use
* Implement data compression for storage

---

## \*\*SECTION 6: PERFORMANCE MONITORING & OPTIMIZATION\*\*

### \*\*6.1 Token Usage Tracking\*\*

:

TOKEN\_METRICS = {  
 "input\_tokens": 0,  
 "output\_tokens": 0,  
 "tool\_call\_tokens": 0,  
 "total\_cost": 0.00,  
 "processing\_time": 0,  
 "cache\_hit\_rate": 0.0  
}

:

* Track token usage per agent invocation
* Monitor cost per analysis type
* Calculate efficiency improvements over time
* Alert when costs exceed thresholds

### \*\*6.2 Performance Benchmarks\*\*

:

|  |  |  |  |
| --- | --- | --- | --- |
| **Metric** | **Baseline** | **Target** | **Measurement** |
| \*\*Token Cost per Analysis\*\* | 100% | 30-50% | Cost tracking |
| \*\*Processing Time\*\* | 100% | 50% | Time measurement |
| \*\*Cache Hit Rate\*\* | 0% | 80% | Cache analytics |
| \*\*Parallel Execution Success\*\* | 70% | 95% | Success rate tracking |

### \*\*6.3 Continuous Optimization Process\*\*

:

1. \*\*Week 1\*\*: Analyze token usage patterns
2. \*\*Week 2\*\*: Identify optimization opportunities
3. \*\*Week 3\*\*: Implement improvements
4. \*\*Week 4\*\*: Measure impact and document lessons learned

---

## \*\*IMPLEMENTATION CHECKLIST\*\*

### \*\*Phase 1: Immediate Optimizations (Week 1)\*\*

* [ ] Implement 40% prompt compression for all agent instructions
* [ ] Enable parallel tool execution in workflows
* [ ] Deploy structured tag format for agent communications
* [ ] Implement basic token usage tracking

### \*\*Phase 2: Intermediate Optimizations (Week 2-3)\*\*

* [ ] Deploy result caching for expensive operations
* [ ] Implement batch processing for data collection
* [ ] Optimize agent selection based on efficiency matrix
* [ ] Deploy template-based report generation

### \*\*Phase 3: Advanced Optimizations (Week 4-6)\*\*

* [ ] Implement comprehensive context sharing optimization
* [ ] Deploy automated workflow orchestration
* [ ] Implement advanced performance monitoring
* [ ] Complete efficiency benchmarking and validation

### \*\*Success Validation\*\*

* [ ] Achieve 40-70% token cost reduction
* [ ] Maintain or improve output quality scores
* [ ] Reduce processing time by 50%
* [ ] Achieve 95% parallel execution success rate

---

## \*\*EMERGENCY PROCEDURES\*\*

### \*\*Token Budget Overruns\*\*

1. \*\*Immediate Actions\*\*: Enable aggressive caching, reduce analysis scope
2. \*\*Short-term\*\*: Review and optimize most expensive operations
3. \*\*Long-term\*\*: Re-evaluate tool selection and agent efficiency

### \*\*Performance Degradation\*\*

1. \*\*Diagnose\*\*: Check token usage patterns and processing times
2. \*\*Optimize\*\*: Focus on highest-impact optimization areas
3. \*\*Monitor\*\*: Implement enhanced tracking for problem areas

---

: Token optimization SOP ready for implementation

: 40-70% cost reduction with maintained quality

: 4-6 weeks for full optimization

---