Forge the Anvil — PI Orchestrator Fix

Plan: Forge the Anvil — PI Orchestrator Fix

Research Summary

Analysed pi-orchestrator.js (1706 lines), ollama-tool-agent.js (548 lines), hivemind.js (994 lines), dag-scheduler.js (394 lines), plus MC stats, minions.db, events.db, and HiveMind data.

Key findings:

- Minion runs: 24 total, 5 completed (21%), 16 cancelled, 3 failed
- Event chain: 62 delegates → 11 completions (18%)
- Proof-of-work bug CONFIRMED: `reportCompletion()` line 1091 checks for GOTCHA/verify dir BEFORE line 1104 creates them
- Budget flat $2.00 for all Claude CLI tasks (line 782)
- ollama-tool-agent: MAX_TURNS=10, no JSON repair, no retry
- HiveMind: 6,843 entries, only 4 types have TTLs — briefing/alert/coordination. Everything else permanent
- dag-scheduler already has `fail()` — no work needed
- routing-outcomes.db exists but tracks routing, not task outcomes

Objective

Fix the 3 highest-impact bugs in PI orchestrator, harden the ollama agent, and add basic outcome tracking. Target: minion completion >50% on next 20+ runs.

Team Orchestration

Team Members

ID	Name	Role	Agent Type
B1	proof-of-work-builder	Fix reportCompletion ordering + budget scaling	builder
V1	proof-of-work-validator	Validate proof-of-work fix + budget	validator
B2	ollama-hardening-builder	Harden ollama-tool-agent	builder
V2	ollama-hardening-validator	Validate ollama hardening	validator
B3	hivemind-ttl-builder	Add TTLs + outcome tracking	builder
V3	hivemind-ttl-validator	Validate HiveMind + tracking	validator

Step-by-Step Tasks

Phase 1: Fix Proof-of-Work Bug + Budget Scaling (HIGHEST IMPACT)

Task 1: Fix reportCompletion proof-of-work ordering

- Owner: B1
- BlockedBy: none
- File: `~/system/kernel/pi-orchestrator.js`
- What: Move the "AUTO QA PREP" block (lines 1104-1146) BEFORE the "VALIDATION GATE" check (lines 1079-1101). The auto-prep creates `/tmp/verify-{id}` and `/tmp/gotcha-task-{id}.md` that the gate checks for. Currently gate checks first → always fails for local models that can't write these files.
- Acceptance:
  - ☐ Auto-prep block executes BEFORE proof-of-work check
  - ☐ Proof-of-work check still blocks if auto-prep produced nothing AND agent produced nothing
  - ☐ `node -c ~/system/kernel/pi-orchestrator.js` passes (syntax valid)

Task 2: Scale Claude CLI budget by complexity

- Owner: B1
- BlockedBy: 1
- File: `~/system/kernel/pi-orchestrator.js` (pipelineClaudeCLI function, line 782)
- What: Replace hardcoded `'--max-budget-usd', '2.00'` with complexity-based budget. Read complexity from task or modelSelection. Map: complexity 1-2=$0.50, 3=$1.00,

4=$2.00, 5=$5.00.
- How: The `pipelineClaudeCLI` function already receives `task` parameter. Add budget lookup before spawn. Pass `task.budget` or compute from modelSelection.tier.
- Acceptance:
  - ☐ Budget varies by tier (check spawn args in code)
  - ☐ Tier 5 gets $5.00, Tier 4 gets $2.00, lower tiers get less
  - ☐ Syntax check passes

Task 3: Add human-active backoff

- Owner: B1
- BlockedBy: 2
- File: `~/system/kernel/pi-orchestrator.js` (runCycle function, around line 1234)
- What: At top of runCycle, check if `/tmp/mc-active-task` exists and is fresh (<5 min). If so, skip Claude CLI tasks (set claude concurrency to 0 for this cycle). Local models still run.
- Acceptance:
  - ☐ When human is active, cloud tasks are deferred
  - ☐ Local model tasks still execute
  - ☐ When marker is stale (>5 min), normal behavior resumes

Task 4: Validate Phase 1 fixes

- Owner: V1
- BlockedBy: 3
- What: Read pi-orchestrator.js and verify all 3 changes. Run syntax check. Verify auto-prep is before gate check. Verify budget is dynamic. Verify human backoff logic.
- Acceptance:
  - ☐ `node -c ~/system/kernel/pi-orchestrator.js` exits 0
  - ☐ Auto-prep block appears BEFORE proof-of-work check in reportCompletion
  - ☐ Budget in pipelineClaudeCLI is NOT hardcoded $2.00
  - ☐ runCycle has human-active check

Phase 2: Harden Ollama Tool Agent

Task 5: Increase MAX_TURNS and add JSON repair

- Owner: B2
- BlockedBy: none (parallel with Phase 1)
- File: `~/system/tools/ollama-tool-agent.js`
- What:
  1. Change `MAX_TURNS = 10` to `MAX_TURNS = 20` (line 44)
  2. In the JSON parse block (around line 320), add fallback: if `JSON.parse(jsonStr)` fails, try regex extraction `/{[\s\S]*}/` and retry parse

3. In tool execution (around line 338-352), wrap in retry: if TOOLS[tool].exec throws, retry once with 2s delay

- Acceptance:
  - ☐ MAX_TURNS is 20
  - ☐ JSON parse has regex fallback before giving up
  - ☐ Tool execution retries once on failure
  - ☐ `node -c ~/system/tools/ollama-tool-agent.js` passes

Task 6: Validate ollama-tool-agent changes

- Owner: V2
- BlockedBy: 5
- What: Read ollama-tool-agent.js. Verify MAX_TURNS, JSON repair, and retry logic.
- Acceptance:
  - ☐ MAX_TURNS = 20
  - ☐ JSON fallback regex exists after parse failure
  - ☐ Tool exec has try/catch with retry
  - ☐ Syntax valid

Phase 3: HiveMind TTL + Outcome Tracking

Task 7: Add aggressive TTLs to HiveMind

- Owner: B3
- BlockedBy: none (parallel with Phase 1 and 2)
- File: `~/system/agents/hivemind/hivemind.js` (TTL_DAYS object, line 201)
- What: Expand TTL_DAYS to cover high-volume types currently permanent:

```
briefing: 3       (was 7)
alert: 14          (was 30)
coordination_request: 7  (unchanged)
coordination_response: 7 (unchanged)
update: 7          (NEW — 306 entries)
info: 7            (NEW — 378 entries)
queue: 3           (NEW — 302 entries)
error: 14          (NEW — 290 entries)
report: 14         (NEW — 284 entries)
task-update: 7     (NEW — 221 entries)
task: 7            (NEW — 116 entries)
spam: 1            (NEW — 99 entries)
intake: 7          (NEW — 94 entries)
```

- Acceptance:
  - ☐ TTL_DAYS has 13+ entries (was 4)
  - ☐ briefing reduced to 3 days
  - ☐ alert reduced to 14 days
  - ☐ spam has 1 day TTL
  - ☐ Syntax valid

Task 8: Add task outcome tracking to pi-orchestrator

- Owner: B3
- BlockedBy: 7
- File: `~/system/kernel/pi-orchestrator.js`
- What: After verifyQuality in runCycle (around line 1293), record outcome to `~/system/databases/learning-loop.db`:
  1. Create SQLite DB + table at top of file (lazy init like other DBs):

```
CREATE TABLE IF NOT EXISTS task_outcomes (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  mc_task_id INTEGER, task_domain TEXT, task_type TEXT,
  complexity INTEGER, tier INTEGER, model TEXT, pipeline TEXT,
  success BOOLEAN, quality_score REAL, response_length INTEGER,
  latency_ms INTEGER, failure_reason TEXT, priority TEXT,
  created_at TEXT DEFAULT (datetime('now'))
);
```

  2. Insert row after quality gate check (both pass and fail paths)
  3. Also insert on pipeline failure (result.success=false path)
- Acceptance:
  - ☐ learning-loop.db gets created on first run
  - ☐ task_outcomes table has correct schema
  - ☐ Outcomes recorded for success AND failure paths
  - ☐ Syntax valid

Task 9: Selective HiveMind feedback

- Owner: B3
- BlockedBy: 8
- File: `~/system/kernel/pi-orchestrator.js` (feedbackToHiveMind function, line 1211)
- What: Currently stores ALL tier 2+ responses >200 chars. Change to:
  1. Only store tier 3+ responses (skip tier 2 — too much noise)
  2. Truncate to 1000 chars (was 5000)
  3. Skip if quality gate failed (no failed responses into HiveMind)
- Acceptance:

- [ ] feedbackToHiveMind checks tier >= 3 (was >= 2)
- [ ] Truncation is 1000 chars (was 5000)
- [ ] Function is only called after quality gate passes

Task 10: Validate Phase 3

- Owner: V3
- BlockedBy: 9
- What: Read hivemind.js and pi-orchestrator.js. Verify TTLs, outcome tracking, and selective feedback.
- Acceptance:
  - [ ] HiveMind TTL_DAYS has 13+ entries
  - [ ] learning-loop.db schema is correct
  - [ ] feedbackToHiveMind only stores tier 3+, truncated to 1000 chars
  - [ ] Both files pass syntax check

Validation Commands

```
# Syntax check all modified files
node -c ~/system/kernel/pi-orchestrator.js
node -c ~/system/tools/ollama-tool-agent.js
node -c ~/system/agents/hivemind/hivemind.js

# Restart daemon
launchctl kickstart -k gui/501/com.john.pi-orchestrator

# Monitor
tail -f ~/system/logs/pi-orchestrator.log

# Check HiveMind cleanup
node ~/system/agents/hivemind/hivemind.js cleanup

# Check learning loop DB
sqlite3 ~/system/databases/learning-loop.db "SELECT COUNT(*) FROM task_outcomes"
```

What This Plan Does NOT Include (Intentionally)

- **Workflow engine** — premature. Fix the basics first.
- **Adaptive router (Thompson Sampling)** — only 24 data points. Need 100+ outcomes before routing optimization makes sense.
- **Project coordinator** — can't orchestrate projects if single tasks fail.
- **Daemon consolidation** — works fine, not broken.
- **Contract test suite** — nice to have but not the bottleneck.

These belong in Phase 2 AFTER Gate 1 metrics prove the foundation works.

Gate 1 Success Criteria

After these fixes, run PI orchestrator for 48h and measure:

- Minion Success Rate > 50% on 20+ new tasks
- No false completions (tasks marked done with no output)
- HiveMind entry count drops after cleanup
- learning-loop.db has outcome data for every task attempt

Revision #3
Created 2026-03-10 16:31:30 UTC by John
Updated 2026-05-31 20:05:01 UTC by John