Strategic Engineering of High-Performance Resource Engines: A Technical Analysis of Remote Mining and Reservation Architectures in Screeps

The transition from a localized, single-room economy to a distributed, multi-room resource engine represents the primary inflection point in a Screeps AI's developmental trajectory. In the deterministic environment of the Screeps game engine, energy acquisition is the fundamental substrate upon which all other operations—military expansion, technological advancement through Global Control Levels (GCL), and market participation—are built. The "Resource Engine," specifically defined here as the integrated system of remote harvesting, room reservation, and long-range logistics, serves as the primary driver of colony growth. While basic harvesting logic suffices for early Room Control Levels (RCL), the implementation of a top-tier resource engine requires a rigorous application of computational geometry, graph theory, and precise mathematical modeling of creep body mechanics and room-level thermodynamics.

Mathematical Foundations of the Harvesting Cycle

The optimization of a resource engine begins with the source. Each energy source in the Screeps world is governed by a 300-tick regeneration cycle. In unreserved rooms, sources provide 1,500 energy units per cycle, translating to a maximum throughput of 5 energy units per tick. The application of a reserveController intent by a creep with one or more CLAIM parts doubles this capacity to 3,000 energy units per cycle, or 10 energy units per tick. This doubling of output is the first and most significant efficiency gain in a remote mining operation.

Harvesting Power and Miner Body Composition

The Screeps engine defines HARVEST_POWER as 2 energy units per tick per WORK part. To achieve 100% harvesting efficiency in a reserved room, a miner must possess exactly 5 WORK parts, yielding 10 energy units per tick. Any surplus of WORK parts beyond this limit provides no additional energy from the source, although it may be utilized for collateral tasks such as the maintenance of local infrastructure.

Mining State	Source Capacity	Regeration Cycle	Required WORK Parts	Net Energy/Tick
Unreserved	1,500	300 Ticks	3 (effective 2.5)	5
Reserved	3,000	300 Ticks	5	10
Source Keeper	4,000	300 Ticks	7 (effective 6.6)	13.33

The efficiency of a miner is defined as the ratio of energy harvested to the energy cost of the creep over its 1,500-tick lifespan. A standard remote miner with the body $$ costs 800 energy. Assuming a travel distance of 50 tiles, the miner arrives at the source with 1,450 ticks of life

remaining. Total energy harvested is calculated as:
Where D is the distance from the spawn, P_{harvest} is HARVEST_POWER, and W is the number of WORK parts. For a 5-WORK miner at 50 tiles, E_{total} = (1450) \times (10) = 14,500 energy units. The net profit of the creep is E_{total} - C_{spawn}, or 13,700 energy units.
In advanced implementations, a 6th WORK part is often added to allow the miner to repair its own container. Since a container in a remote room decays by 5,000 hits every 100 ticks, it requires 50 energy units of repair every 100 ticks. One WORK part repairs 100 hits per tick at the cost of 1 energy unit. Therefore, the miner can spend 1 tick out of every 20 repairing the container to maintain the structure indefinitely, eliminating the need for a dedicated repair creep and reducing CPU overhead by centralizing tasks.

The Physics of Logistics: Transportation and Fatigue

The most complex component of the resource engine is the transportation of harvested energy back to the home room. Unlike harvesting, which is capped by source regeneration, hauling efficiency is a variable of distance, creep body composition, and terrain type.

The Fatigue Equation and Movement Optimization

Fatigue is the primary constraint on hauler throughput. The engine generates fatigue based on the following formula:
However, CARRY parts only generate fatigue when they contain resources. An empty hauler moves at a speed of 1 tile per tick regardless of its MOVE to CARRY ratio, provided it has at least one active MOVE part. A loaded hauler requires one MOVE part for every non-MOVE part to maintain a speed of 1 tile per tick on plain terrain (V = 1 \text{ tile/tick}). On roads, the fatigue generation is halved, allowing a 1:2 ratio of MOVE to CARRY parts to maintain maximum velocity.

Terrain Type	Fatigue per Part	MO[span_47](start_span)[span_47](end_span)[span_54](start_span)[span_54](end_span)VE Ratio Required	Effective Speed (Max)
Road	1	1:2	1 tile/tick
Plain	2	1:1	1 tile/tick
Swamp	10	5:1	1 tile/tick

The total "hauling power" required for a source is the product of the energy generated per tick and the round-trip distance. For a reserved source (10 \text{ e/tick}) at a distance of 50 tiles, the round-trip distance is 100 tiles. To prevent energy from accumulating and decaying on the ground, the hauler pool must move 1,000 energy units every 100 ticks.

Part-Count Balancing vs. Headcount

Top-tier resource engines utilize "Part-Count Balancing" to determine hauler requirements. Instead of spawning a fixed number of creeps, the engine calculates the total CARRY parts needed:
Where E_{gen} is energy per tick and D is distance. For the 50-tile example, 20 CARRY parts are required. At RCL 3, where energy capacity is 800, a creep can support a maximum of 10

CARRY and 5 MOVE parts (costing 750 energy). Thus, the engine spawns two haulers. At RCL 8, the same demand is met by a single, more efficient creep with 20 CARRY and 10 MOVE parts, reducing CPU costs associated with pathfinding and intent processing.

Room Reservation and Sovereignty

Reservation is the mechanism that transitions a remote room from a temporary resource site to a stable extension of the colony's economy. Beyond the 100% increase in energy yield, reservation provides a critical defensive benefit: it prevents NPC invaders from spawning at exits leading to the reserved room.

The Economy of the CLAIM Part

The CLAIM part is the most expensive and specialized body part in the game, costing 600 energy and reducing the creep's lifetime from 1,500 to 500 or 600 ticks. A reserver body is typically [1 \text{ CLAIM}, 1 \text{ MOVE}], costing 650 energy.
The reservation buffer on a controller can store up to 5,000 ticks. Each reserveController action adds N ticks to the buffer, where N is the number of CLAIM parts, and the buffer decays by 1 tick per game tick. To increase the reservation level, a creep must have at least 2 CLAIM parts, as 1 part merely offsets the natural decay.

Strategy	CLAIM Parts	Net Buffer Gain	Ticks to Max Buffer
Maintenance	1	0 ticks/tick	Constant
Buffer Building	2	1 tick/tick	5,000 Ticks
Rapid Reset	5	4 ticks/tick	1,250 Ticks

A top-tier engine avoids constant reserver presence. Instead, it employs "Buffer Cycling." The engine tracks the ticksToEnd of the reservation. When the buffer falls below a threshold defined as D + T_{spawn} + T_{safety} (where D is travel distance, T_{spawn} is time to spawn, and T_{safety} is a buffer for congestion), the engine pushes a reserver request to the spawn queue. This minimizes the energy spent on the expensive 600-tick lifetime of reservers.

Architectural Forensics: The Overmind Framework

Modernization of a resource engine necessitates a shift from role-based logic to an objective-based architectural framework, such as the "Overmind" or "Boardroom" models. Role-based systems, where each creep independently decides its action (e.g., "if I am a harvester, I find a source"), suffer from poor scalability and inefficient resource allocation.

Directives and Overlords

The Overmind architecture organizes the empire into a hierarchy of functional objects:

1. **Colony:** The top-level object wrapping a single owned room and its associated remote territories.
2. **Directive:** A persistent object (often associated with a flag) that identifies a strategic objective in the game world, such as RemoteMiningDirective or GuardDirective.
3. **Overlord:** A process instantiated by a Directive to handle the fulfillment of the objective. The MiningOverlord calculates the necessary body parts for miners and haulers, monitors their life-cycles, and issues spawn requests.
4. **Zerg/Minions:** Creeps that are stripped of independent logic and instead act as executors

of Tasks assigned by the Overlord.

This structure allows the engine to treat remote mining as a "straightforward optimization problem". When a RemoteMiningDirective is placed in a room, the associated Overlord analyzes the room’s topography, identifies the sources, calculates the distances to the home-room storage, and determines the optimal part counts for the logistics loop.

The Logistics Network and Transport Requests

Fulfillment of energy transport is handled by a global LogisticsNetwork. Instead of a hauler being "tethered" to a specific miner, it responds to TransportRequests.

- **Output Requests:** A container at a remote source reaching a capacity threshold (e.g., 1,600/2,000 units) generates an output request.
- **Input Requests:** A tower needing 200 energy or a spawn needing 300 energy generates an input request.
- **Priority Scoring:** The LogisticsNetwork scores requests based on distance and urgency. A hauler is assigned to pick up resources from a remote source and is then dynamically reassigned to the highest priority input request in the home room.

This "task-chaining" allows haulers to maintain high duty cycles. For example, a hauler returning from a remote room might be assigned to fill a nearby tower before returning to its "idle" state at the storage, maximizing the energy moved per CPU tick.

Defensive Protocols and NPC Management

Remote rooms are the primary engagement zone for NPC Invaders and Source Keepers. A high-performance resource engine must integrate defensive automation directly into its logistics cycle.

NPC Invader Spawn Logic

Invaders spawn based on an internal counter that tracks the total energy harvested in a room, typically triggering after 100,000 energy units are extracted. These invaders come in two tiers: "Light" creeps in rooms below RCL 4 and "Heavy" creeps in higher-level rooms.

Top-tier engines implement a multi-stage response:

1. **Detection:** The engine scans remote rooms for FIND_HOSTILE_CREEPS. If an invader is detected, a "State of Emergency" is declared for that room.
2. **Evacuation:** All non-combat creeps (miners and haulers) are ordered to flee to a safe room to prevent unnecessary life-loss and energy drop-off.
3. **Dynamic Response Spawning:** The DefenseOverlord analyzes the invader’s body. If the invader is a melee attacker, the engine spawns a ranged kiter. If it is a squad with a healer, the engine spawns a high-damage melee attacker or a superior ranged-mass-attack creep.
4. **Tombstone Reclamation:** After the invader is destroyed, the engine assigns a hauler to withdraw the energy and boosts from the invader's tombstone, ensuring the engagement remains energy-positive.

Source Keeper Room Operations

Source Keeper (SK) rooms are high-risk, high-reward environments providing 4,000 energy per source. Mining these rooms requires "Keeper Suppression." A combat creep, typically with a high HEAL and ATTACK count, is stationed in the room to kill Source Keepers as they spawn from their Lairs every 300 ticks.

Role	SK Miner	SK Hauler	SK Guard
Body	7 WORK, 1 CARRY, 4 MOVE	20 CARRY, 10 MOVE	10 ATTACK, 5 HEAL, 15 MOVE
Function	Static harvest and repair	Long-range transport	Suppression of Lairs

The introduction of InvaderCores adds a new layer of complexity. These cores can launch "Lesser Cores" into remote rooms, which reserve the controller and prevent harvesting until destroyed. A top-tier engine must detect these cores and spawn a "Cleaner" creep—a simple body with ATTACK or WORK parts—to dismantle the core and restore harvesting operations.

Traffic Management and Pathing Optimization

Traffic is the "silent killer" of remote mining efficiency. In a single-lane road system, one slow creep can delay the entire logistics chain, leading to source overflow and decay.

Movement Libraries and Caching

Top-tier bots utilize movement libraries like Traveler or Cartographer to replace the default creep.moveTo(). These libraries offer:

- **Path Caching:** Paths are stored in Heap or Memory to avoid expensive PathFinder calls every tick.
- **Hostile Room Avoidance:** The pathfinder automatically routes around rooms that have been flagged as dangerous.
- **Traffic Shoving:** Idle creeps are ordered to move out of the way of moving creeps.

The Bipartite Matching Algorithm for Traffic

The gold standard for traffic management is the Bipartite Matching Problem (BMP). Every tick, the engine collects move intents from all creeps in a room.

1. **Intent Mapping:** Creeps express a desire for a target tile or a preference to stay put.
2. **Graph Construction:** The engine builds a graph connecting creeps to potential tiles.
3. **Resolution:** The engine resolves collisions by "shoving" lower-priority creeps (like upgraders or stationary miners) into empty adjacent tiles to allow higher-priority haulers to maintain their velocity.

This ensures that even in narrow corridors, creeps can "swap" positions in a single tick, maintaining a constant flow of resources.

Infrastructure: Road Logistics and Container Management

The decision to build roads is a purely economic one. The cost to repair a road on plain terrain is 1 energy per 1,000 ticks. The cost of the extra MOVE parts required to move at the same speed without a road is significantly higher over the life of the creep.

The Road-Repair-on-Transit Pattern

To maintain hundreds of tiles of remote roads, the most efficient method is attaching one WORK part to haulers.

- **Mechanism:** As the hauler moves back and forth, it checks the hits of the road tile it is standing on.
- **Action:** If the road is below 100% hits, the hauler issues a repair intent.
- **Result:** This keeps the road network at full health without the CPU or energy cost of a dedicated repairer.

Container Placement and Static Harvesting

Containers are essential for "Static Harvesting," where the miner never moves. The container should be placed on the tile adjacent to the source that offers the best pathing to the room exit.

- **Decay:** Containers in remote rooms decay at 5,000 hits per 100 ticks.
- **Repair Cost:** Maintenance costs 0.5 energy per tick (50 \text{ energy} / 100 \text{ ticks}).
- **Dropped Energy:** If no container is used, dropped energy decays at \lceil \text{amount} / 1000 \rceil per tick. At a reserved source producing 3,000 energy, the decay of dropped energy significantly exceeds the repair cost of a container, making containers mandatory for top-tier efficiency.

Economic Analysis: Return on Investment (ROI) and Scaling

A resource engine must be self-evaluating. Top-tier implementations track the "Energy Per Tick" (EPT) of every remote room to determine which targets are most profitable.

Efficiency Metrics

The efficiency of a remote room is defined as:
Where E_{max} is 10 energy/tick for reserved rooms.

- At 50 tiles, efficiency is approximately 65%.
- At 100 tiles, efficiency drops to 52%.
- Beyond 250 tiles, the cost of the haulers and the road decay often results in negative efficiency, marking the "Economic Horizon" of the colony.

Scaling with Power Creeps and Boosts

At high RCL, the resource engine can be further optimized using PowerCreeps and lab boosts.

- **EXTEND_SOURCE:** A Power Creep can increase the capacity of a source beyond 3,000, allowing for massive energy spikes.
- **REGEN_SOURCE:** Accelerates the 300-tick regeneration cycle.
- **WORK Boosts:** Compounds like UO2 (Utrium Oxide) increase harvest effectiveness, allowing a miner to exhaust a source with fewer body parts, thereby saving on spawn energy and CPU.

Modernization Roadmap: Implementing the Top-Tier Engine

Based on the forensic analysis of standard Screeps architectures and the provided documentation, the following steps are required to implement a top-tier resource engine.

Phase 1: The Logistics Backbone

1. **Implement the Logistics Network:** Move energy handling to a request-based system where haulers respond to TransportRequests rather than being hard-coded to sources.
2. **Part-Count Balancing:** Replace "headcount" logic with a system that calculates the total CARRY parts needed for a route and spawns the minimum number of creeps to fulfill that need.
3. **Road Automation:** Implement the "Repair-on-Transit" logic for haulers to maintain infrastructure automatically.

Phase 2: The Overlord Migration

1. **Refactor into Directives:** Use flag-based directives to identify remote mining targets.
2. **Instantiate Overlords:** Create a MiningOverlord to manage the life-cycles of remote creeps and handle local state transitions (e.g., switching from "Mining" to "Evacuating" during an invasion).
3. **Buffer-Based Reservation:** Implement reservation logic that maintains the 5,000-tick buffer with minimal reserver uptime.

Phase 3: Traffic and Pathing Sovereignty

1. **Integrate Cartographer/Traveler:** Replace moveTo with a cached pathfinder that supports traffic management and hostile room avoidance.
2. **Implement BMP Traffic Management:** Use graph-based matching to resolve creep collisions in high-traffic corridors.
3. **CostMatrix Optimization:** Dynamically update CostMatrices to account for road decay and stationary creeps, ensuring haulers always take the most efficient route.

Phase 4: Defensive and Offensive Integration

1. **Automate Invader Responses:** Build a DefenseOverlord that calculates specific counter-bodies for NPC invaders based on their detected parts.
2. **SK Suppression Logic:** Implement the "Lair-Cycling" guard for Source Keeper rooms to maximize high-yield energy extraction.
3. **Invader Core Cleanup:** Automate the detection and destruction of Inva[span_130](start_span)[span_130](end_span)derCores to prevent sovereignty loss in remote territories.

Conclusion: The Integrated Resource Engine

The development of a high-performance resource engine is not a singular task but an ongoing engineering effort to balance energy throughput against CPU and spawn-time costs. By applying the mathematical rigors of body-part balancing, adopting the modularity of the Overmind architecture, and implementing sophisticated traffic management, an AI can achieve a level of economic stability that supports sustained expansion. In the competitive environment of Screeps, the resource engine is the heart of the colony; its efficiency determines the ultimate survival and dominance of the empire. Through the implementation of these best practices, the engineer moves from manual management to a state of autonomous economic sovereignty.

Works cited

1. Workflow tips and prioritization for new players? | Screeps Forum, https://screeps.com/forum/topic/2556/workflow-tips-and-prioritization-for-new-players 2. Resources | Screeps Documentation, https://docs.screeps.com/resources.html 3. Remote mining article by slowmotionghost · Pull Request #60 · screeps/docs - GitHub, https://github.com/screeps/docs/pull/60/files 4. Great Filters - Screeps Wiki, https://wiki.screepspl.us/Great_Filters/ 5. Remote Harvesting - Screeps Wiki, https://wiki.screepspl.us/Remote_Harvesting/ 6. Screeps #1: Overlord overload | Ben Bartlett, https://bencbartlett.com/blog/screeps-1-overlord-overload/ 7. Journey to Solving the Traffic Management Problem - Harabi Screeps, https://sy-harabi.github.io/Journey-to-Solving-the-Traffic-Management-Problem/ 8. Reservation - Screeps Wiki, https://wiki.screepspl.us/Reservation/ 9. This is the slowest paced game ever. Does it get more interesting once GCL goes up? : r/screeps - Reddit, https://www.reddit.com/r/screeps/comments/8181nc/this_is_the_slowest_paced_game_ever_does_it_get/ 10. Creep | Screeps Wiki | Fandom, https://screeps.fandom.com/wiki/Creep 11. Is there a guide out there for the ideal ratios of your creeps? : r/screeps - Reddit, https://www.reddit.com/r/screeps/comments/864qy3/is_there_a_guide_out_there_for_the_ideal_ratios/ 12. Faster Controller Levelling : r/screeps - Reddit, https://www.reddit.com/r/screeps/comments/5jia18/faster_controller_levelling/ 13. Questions on roles | Screeps Forum, https://screeps.com/forum/topic/913/questions-on-roles 14. Energy - Screeps Wiki, https://wiki.screepspl.us/Energy/ 15. What's the function to calculate creep cost? | Screeps Forum, https://screeps.com/forum/topic/2288/what-s-the-function-to-calculate-creep-cost 16. NPC Invaders | Screeps Documentation, https://docs.screeps.com/invaders.html 17. CLAIM body parts and 'time to live' : r/screeps - Reddit, https://www.reddit.com/r/screeps/comments/6pz7ij/claim_body_parts_and_time_to_live/ 18. CLAIM part too inefficient | Screeps Forum, https://screeps.com/forum/topic/1167/claim-part-too-inefficient 19. Respawning - Screeps Documentation, https://docs.screeps.com/respawn.html 20. Screeps #5: Refactoring for Remote Mining - Field Journal, https://jonwinsley.com/notes/screeps-refactoring-remote-mining 21. How i can control second room? | Screeps Forum, https://screeps.com/forum/topic/1377/how-i-can-control-second-room 22. Releases · bencbartlett/Overmind - GitHub, https://github.com/bencbartlett/Overmind/releases 23. Overmind/src/overlords/Overlord.ts at master · bencbartlett/Overmind - GitHub,

https://github.com/bencbartlett/Overmind/blob/master/src/overlords/Overlord.ts 24. Source keeper...what is that? | Screeps Forum, https://screeps.com/forum/topic/1533/source-keeper-what-is-that 25. Invader - Screeps Wiki, https://wiki.screepspl.us/Invader/ 26. Keeper Lairs and Invader Swarms Strategy | Screeps Forum, https://screeps.com/forum/topic/327/keeper-lairs-and-invader-swarms-strategy 27. Invader Cores suck. No Invader Core Shard | Screeps Forum, https://screeps.com/forum/topic/2823/invader-cores-suck-no-invader-core-shard 28. Traffic Management | screeps-cartographer, https://glitchassassin.github.io/screeps-cartographer/pages/trafficManagement.html 29. Cartographer is an advanced (and open source) movement library for Screeps - GitHub, https://github.com/glitchassassin/screeps-cartographer 30. bonzaiferroni/Traveler: Traveler - A general movement solution for Screeps.com - GitHub, https://github.com/bonzaiferroni/Traveler 31. screeps-cartographer, https://glitchassassin.github.io/screeps-cartographer/ 32. Screeps #27: Optimizing Pathfinding with Rust | Field Journal, https://jonwinsley.com/notes/screeps-clockwork 33. Power Creeps update | Screeps Forum, https://screeps.com/forum/post/10564