Aurora AI Technical Specs

1. System Overview

Aurora AI is a family of large language models (LLMs) optimized for:

- Multi-step tool use (MCP-style tools, REST APIs, vector DBs)
- Long-context reasoning and retrieval over heterogeneous documents
- Agent-style workflows (planning, reflection, multi-turn task execution)

The current flagship model is **Aurora-72B-Agent-v3**, a 72-billion parameter decoder-only transformer trained with:

- Mixed open + synthetic + curated proprietary-style corpora
- Multi-stage instruction tuning and RLHF
- Tool-usage finetuning for structured tool calling

2. Model Architecture

2.1 Core Architecture

- **Model family:** Decoder-only Transformer (GPT-style)
- **Parameters:** 72 billion (non-embedding parameters)
- **Layers:** 96 transformer blocks
- **Hidden size:** 8192
- **Attention heads:** 64 (per layer)
- **Head dimension:** 128
- **Feed-forward dimension:** 4 × hidden = 32,768 with gated activation (GEGLU)
- **Positional encoding:** Rotary Position Embeddings (RoPE) extended for 256k context via NTK scaling
- **Normalization:** Pre-layernorm with RMSNorm
- **Activation:** SwiGLU / GEGLU hybrid depending on layer (alternating)
- **Attention variant:**
    - Multi-Query Attention (MQA) for KV cache compression
    - FlashAttention v2-style fused kernels for training and inference
    - Sliding window + global tokens for efficient long-context

2.2 Context Length & Memory

- **Max context length:** 256,000 tokens
- **KV cache quantization:**
    - Inference KV cache stored in 8-bit FP8-like format
    - Per-layer dynamic scaling factors

- **Long-context strategy:**
  - Hybrid attention: full attention for first 8k tokens, windowed attention (4k) beyond, with sparse global tokens every 512 tokens
  - Retrieval-augmented chunk injection via side-channel (see §5)

2.3 Tokenization

- **Tokenizer type:** Unigram/BPE hybrid
- **Vocab size:** 65,536 tokens
- **Script coverage:** Latin, CJK, Cyrillic, Devanagari, Arabic, basic emoji, code tokens
- **Special tokens:**
  - <tool_call>, <tool_result>, <sys>, <user>, <assistant>
  - <agent_plan>, <agent_reflection> for internal agent states
- **Byte-fallback:** Enabled for robustness to unknown scripts/binary blobs

3. Training Data & Data Engineering

3.1 Data Mixture (Pretraining)

Approximate effective token counts (after deduplication & filtering):

- **General web crawl:** ~900B tokens
- **Curated technical content:** ~300B tokens
  - APIs, SDK docs, RFCs, scholarly preprints, Q&A, spec documents
- **Code corpora:** ~250B tokens
  - Multilingual: Python, JS/TS, Go, Rust, Java, C/C++, Bash, etc.
- **Conversation data:** ~200B tokens
  - Multi-turn dialogs, forum threads, support transcripts (synthetic/fictive)
- **Math & reasoning:** ~100B tokens
  - Step-by-step solutions, competition problems, proofs
- **Tool-usage synthetic logs:** ~50B tokens
  - Generated via earlier Aurora versions + teacher models

Total effective pretraining mixture: **~1.8T tokens**.

3.2 Data Quality Pipeline

Key pipeline stages:

1. **Ingestion & Normalization**
   - Convert all sources to UTF-8, normalize whitespace and Unicode
   - Strip HTML, preserve code blocks and tables using heuristic rules
1. **Deduplication**
   - *Near-dedup*: SimHash + MinHash-based chunk deduplication

- Per-document shingling (e.g., 13-grams) with Jaccard threshold
- Dedup within source and across sources

1. **Filtering & Safety**
   - Classifiers for: PII, hate, explicit content, spam, templated SEO text
   - Blocklists for high-risk domains / content categories
   - Heuristic filters: short boilerplate, auto-generated junk, mirrored repos
1. **Data Balancing**
   - Target distributions for domains (e.g., at most 25% code, 15% conversational)
   - Temperature sampling / mixture re-weighting per domain
   - Over-sampling underrepresented languages and domain-specific sources
1. **Data Sharding & Curriculum**
   - **Phase 1 (Foundation):** Heavier on general web + code
   - **Phase 2 (Specialized):** Shift toward technical + reasoning corpora
   - **Phase 3 (Refinement):** Higher-quality, curated, filtered subsets

4. Training Regimen

4.1 Hardware & Infrastructure

- **Hardware:**
  - 2048 GPUs (A100 80GB or H100 equivalent) in multi-node clusters
  - NVLink + Infiniband interconnects
- **Parallelism strategies:**
  - Data parallelism (DP) via FSDP / ZeRO-3 style sharding
  - Tensor parallelism (TP) within attention and MLP layers
  - Pipeline parallelism (PP) over transformer blocks
- **Precision:**
  - Mixed-precision training (bfloat16 activations, fp32 master weights)
  - Selective fp32 for layernorm and loss computation

4.2 Optimization

- **Optimizer:** AdamW
  - $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\varepsilon = 1e\text{-}8$
  - Weight decay = 0.1
- **Learning rate schedule:**
  - Warmup for first 3% of steps
  - Cosine decay to 10% of peak LR
  - Peak LR ~1.2e-4 for pretraining, ~5e-5 for finetunes
- **Batching:**
  - Effective global batch size ~6M tokens/step
  - Dynamic padding/truncation per sequence

- Gradient accumulation to reach target batch size

4.3 Finetuning Phases

Aurora's post-pretraining refinement is done in stages:

1. **Instruction Tuning (SFT):**
   - ~5M instruction-response pairs
   - Sources: curated Q&A, synthetic tasks, code edits, tool calls
   - Objective: Supervised next-token loss on assistant outputs
1. **Tool-Use & Agent Tuning:**
   - Focused on sequences of: *plan → tool_call → tool_result → answer*
   - Synthetic interaction logs created by teacher agents and rule-based planners
   - Emphasis on:
     - API schema reading
     - JSON arguments construction
     - Error handling and tool retries
1. **Dialogue Safety & Alignment Tuning:**
   - Additional SFT on safe, policy-compliant responses
   - Red-teaming datasets, refusal patterns, and safety justifications
1. **RLHF + DPO / ORPO:**
   - Human raters compare pairs of outputs (helpfulness vs. safety)
   - Train a reward model on comparison data
   - Optimize with:
     - Classic PPO on shorter sequences
     - DPO / ORPO on larger batched preference sets
   - Constraints encourage:
     - Truthfulness
     - Step-by-step reasoning
     - Tool use when appropriate, not overused

5. Agent & Tool-Use Capabilities

5.1 Tool Calling Interface

Aurora is trained on a structured tool-call protocol:

- **Schema format:** JSON-based tool definitions (akin to OpenAPI / MCP)
- **Invocation format:**

```
{

  "tool_name": "weather_api",

  "arguments": { "location": "Denver, CO", "date": "2025-11-22" }
```

```
}
```

- **Tokens reserved** to delineate tool segments:
    - `<tool_call>{...}</tool_call>`
    - `<tool_result>{...}</tool_result>`

5.2 Planning & Reflection

Aurora uses *internal* planning prefixes (not always shown to user):

- `<agent_plan>` blocks with:
    - Task decomposition
    - Step-by-step subgoals
    - Mapping from subgoals → tools
- `<agent_reflection>` blocks with:
    - Error analysis when tool calls fail
    - Self-check for hallucination risk
    - Plan updates and corrections

These behaviors are induced via:

- Synthetic datasets where teacher agents explicitly produce plan/reflect segments
- RLHF rewards that favor solutions with explicit reasoning and self-checks

6. Evaluation & Benchmarking

6.1 Standard Benchmarks (Fictional Values)

Category	Benchmark	Aurora-72B Score	Comparison Notes
General QA	MMLU-like	83–85%	Strong on STEM + humanities
Coding	HumanEval-like	92%	Robust multi-file reasoning
Math	GSM8K-like	92–94%	Chain-of-thought strongly improves
Long Context	Needle-in-Haystack	98% @ 128k	Retrieval reliable up to 200k tokens
Tool Use	ToolBench-like	89–91%	High accuracy on JSON arguments
Safety/Harms	Internal red-team	Pass > 97%	Refuses on high-risk queries reliably

(All numbers are illustrative / fictional and tuned for documentation.)

6.2 Agent & Tool-Use Benchmarks

Specialized evaluation suites include:

- **Tool Routing Accuracy:**

    - Given a set of candidate tools, pick the correct one
    - Aurora: ~94% micro-accuracy
- **Argument Validity:**
    - Percentage of tool calls with syntactically valid JSON and valid parameter types
    - Aurora: ~98% valid, 1–2% recoverable with retry
- **End-to-End Task Success:**
    - Multi-step workflows (e.g., "search docs → summarize → generate action plan")
    - Success measured by human raters
    - Aurora: ~86–88% task completion without human intervention

6.3 Ablation Studies (Research)

Key ablations conducted during development:

1. **Context Length Ablation**
    - Compare 32k vs 128k vs 256k context models
    - Finding: 128k → 256k yields diminishing returns on general tasks, but large gains on niche long-doc tasks (e.g., legal, codebase comprehension). Production choice: 256k for Aurora-72B, 64k for smaller variants.
1. **Planning Tokens Ablation**
    - Removing <agent_plan> and <agent_reflection> tokens yields:
        - +5–7% hallucination rate on tool-related tasks
        - −3–5% task completion on multi-step workflows
    - Conclusion: plan+reflect tokens significantly benefit agentic reliability.
1. **Tool-Use Data Volume Ablation**
    - Trained models with 25%, 50%, 75%, 100% of tool-use dataset
    - Tool-call correctness scales roughly log-linearly with data amount;
    - 75% of dataset needed to reach >90% argument correctness.
1. **RLHF vs. Pure SFT**
    - Aurora-SFT only vs Aurora-RLHF
    - RLHF model:
        - Reduces unsafe completions by ~60%
        - Increases human-rated helpfulness by ~15–20%
        - Slightly more verbose but within acceptable bounds

7. Safety, Alignment, and Policy Constraints

7.1 Safety Layers

Aurora uses a *multi-layer safety stack*:

1. **Training Data Filtering** (see §3.2)
2. **Policy-Tuned Decoding**

- Safety classifiers run on partial generations (streaming)
- When risk detected, model steered to refuse or redirect

1. **Post-Processing Guards**
    - Deterministic rules for obviously disallowed outputs
    - Output sanitization (e.g., removing leaked API keys, PII patterns)

7.2 Refusal Behavior

Aurora is trained to:

- Decline to provide:
    - Detailed instructions for violence, self-harm, cybercrime
    - Highly sensitive PII reconstruction
- Provide:
    - High-level, non-actionable information when possible
    - Supportive language and safe resources for self-harm content
- Explain:
    - *Why* it is refusing, in simple language
    - Safer alternatives / high-level advice

8. Inference, Serving, and Deployment

8.1 Model Variants

- **Aurora-72B-Agent-v3:** Full capability agent model
- **Aurora-16B-Assistant:** Mid-size assistant model, 128k context
- **Aurora-3B-Edge:** Lightweight, 16k context, on-device/edge optimized

8.2 Quantization & Latency

- **Supported formats:**
    - fp16, bf16, int8, int4
- **Typical deployment configuration (cloud):**
    - 4× H100-equivalent GPUs per replica
    - Int8 weight + KV quantization
    - p95 latency:
        - ~300–600ms for first token
        - ~10–20 tokens/sec generation throughput

8.3 Orchestration

- **Serving stack:**
    - gRPC / HTTP API
    - Request batching and KV cache reuse
    - Per-tenant rate limiting and token quotas

- **Observability:**
  - Metrics: tokens/sec, latency, error rates, refusal stats
  - Traces: per-request tool calls, retries, and user-visible responses
  - Logs: sampled request/response pairs (with redaction) for offline analysis

9. Continuous Training & Monitoring

9.1 Online Improvement Loop

Aurora's lifecycle includes:

1. **Data Collection**
   - Opt-in user interactions (with consent and redaction)
   - Agent failures, tool errors, and escalations
1. **Triage & Labeling**
   - Human review for:
     - Safety violations
     - Hallucinations
     - Poor tool usage
1. **Dataset Curation**
   - Build new SFT and preference datasets from real failures
1. **Periodic Updates**
   - Monthly or quarterly SFT + DPO/RLHF refresh
   - Regression testing across benchmark suites and red-team scenarios

9.2 Drift Detection

- **Distribution shift detectors** on:
  - Input topics / domains
  - Output safety classifier scores
- **Trigger thresholds:**
  - If unsafe output rate exceeds baseline by >X%
  - If hallucination-specific metrics spike in certain domains
- **Mitigations:**
  - Temporary stricter safety settings
  - Blocking specific prompts or patterns
  - Targeted finetunes and policy updates

10. Known Limitations & Future Work

10.1 Current Limitations

- **Residual hallucinations:**
  - Especially in low-resource domains or niche factual queries

- **Limited multimodal capability:**
  - Current Aurora-72B version is text-only; no native image/audio input
- **Tool-dependency brittle in edge cases:**
  - Poorly specified tool schemas or inconsistent tool responses can degrade performance
- **Computational cost:**
  - 72B parameters remain expensive to serve at massive scale

10.2 Future Research Directions

1. **Multimodal Aurora**
   - Extend architecture to support images, audio, and structured sensor data
1. **More Robust Agentic Control**
   - Formal verification-style checks on plans for critical tasks
   - Better isolation between planning tokens and user-visible content
1. **Adaptive Compression**
   - Dynamic layer-skipping / MoE routing for cheaper inference
1. **Stronger Long-Context Retrieval**
   - Jointly trained retriever + generator for 1M+ token contexts
1. **Formal Safety Guarantees**
   - Combining neural models with rule-based or constrained decoders
   - Research into provable bounds on certain risk categories