JOJI Oi Operations Guide

Overview

This document provides operational guidance for the JOJI Oi memory system, including deployment, monitoring, troubleshooting, and maintenance procedures.

System Architecture

Core Components

1. **MemoryAgent**: Thread-safe memory storage with WAL support
2. **ChatMemory**: Conversation management and context handling
3. **WriteAheadLog (WAL)**: Transaction logging for data consistency
4. **BackupManager**: Automated backup and recovery system
5. **Monitoring**: Prometheus metrics and structured logging
6. **FileLock**: Concurrent access control

Data Flow

```
User Input → ChatMemory → MemoryAgent → WAL → File Storage
                ↓              ↓          ↓
            Session Buffer  Validation  Backup
```

Deployment

Prerequisites

- Python 3.9 or higher
- Sufficient disk space for memory storage and backups
- Network access for metrics export (optional)

Installation

Production Deployment

```
# Install from PyPI
pip install jojiai

# Or install from source
git clone https://github.com/gtsurkav-sudo/JOJIAI.git
cd JOJIAI
pip install -e .
```

Docker Deployment

```bash
# Build image
docker build -t jojiai:latest .

# Run container
docker run -d \
  --name jojiai \
  -p 8000:8000 \
  -p 8001:8001 \
  -v /path/to/data:/app/data \
  -v /path/to/logs:/app/logs \
  jojiai:latest
```

Configuration

Environment Variables

- `JOJIAI_MEMORY_PATH` : Path to memory storage (default: `./memory`)
- `JOJIAI_BACKUP_PATH` : Path to backup storage (default: `./backups`)
- `JOJIAI_LOG_LEVEL` : Logging level (default: `INFO`)
- `JOJIAI_METRICS_PORT` : Metrics server port (default: `8000`)
- `JOJIAI_MAX_MEMORY_SIZE` : Maximum memory size in bytes (default: `100MB`)

Configuration File

```json
{
  "memory_path": "/app/data/memory",
  "backup_path": "/app/data/backups",
  "wal_path": "/app/data/wal",
  "max_memory_size": 104857600,
  "backup_interval": 3600,
  "wal_flush_interval": 60,
  "lock_timeout": 30
}
```

Monitoring

Metrics

The system exposes Prometheus metrics on port 8000 by default:

Key Metrics

- `jojiai_operations_total` : Total number of operations by type and status
- `jojiai_operation_duration_seconds` : Operation duration histogram
- `jojiai_memory_usage_bytes` : Memory usage by type
- `jojiai_errors_total` : Total errors by type and component
- `jojiai_retries_total` : Total retries by operation type
- `jojiai_concurrent_operations` : Current concurrent operations

Health Checks

- **Endpoint**: `http://localhost:8000/health`
- **Response**: JSON with system health status

```
{
  "overall_status": "healthy",
  "checks": {
    "memory_agent": {"status": "healthy", "last_check": 1694123456},
    "wal_system": {"status": "healthy", "last_check": 1694123456},
    "backup_system": {"status": "healthy", "last_check": 1694123456}
  },
  "timestamp": 1694123456
}
```

Logging

Structured Logging

All logs are output in JSON format for easy parsing:

```
{
  "timestamp": "2023-09-08T12:00:00Z",
  "level": "INFO",
  "logger": "jojiai.memory_agent",
  "message": "Dialogue stored successfully",
  "dialogue_id": "dialogue_1694123456789",
  "user_id": "user123"
}
```

Log Levels

- `DEBUG`: Detailed debugging information
- `INFO`: General operational messages
- `WARNING`: Warning conditions
- `ERROR`: Error conditions
- `CRITICAL`: Critical system failures

Alerting

Alert Rules

1. **High Error Rate**: Error rate > 5% over 5 minutes
2. **Memory Usage**: Memory usage > 90% of limit
3. **WAL Size**: WAL file size > 100MB
4. **Backup Failure**: Backup operation failed
5. **Lock Timeout**: File lock timeout > 30 seconds

Alert Channels

- Slack notifications
- Email alerts
- PagerDuty integration
- Webhook notifications

Command Line Tools

memoryctl

The `memoryctl` command provides administrative functions:

Basic Commands

```
# Show system status
memoryctl status

# Create backup
memoryctl backup --name manual-backup-$(date +%Y%m%d)

# Restore from backup
memoryctl restore snapshot_20230908_120000 --confirm

# List available backups
memoryctl list-snapshots

# View chat history
memoryctl chat-history --limit 20 --role user

# Search messages
memoryctl search "error message"

# Create conversation summary
memoryctl summarize --count 50

# Cleanup old data
memoryctl cleanup --keep 10

# Validate system integrity
memoryctl validate
```

Advanced Usage

```
# Backup with custom path
memoryctl --memory-path /custom/path backup

# Verbose output
memoryctl -v status

# Search with filters
memoryctl search "python" --limit 5

# Cleanup with specific retention
memoryctl cleanup --keep 5
```

Troubleshooting

Common Issues

1. Race Condition Errors

Symptoms: `ConcurrencyError` exceptions in logs

Causes:
- High concurrent load
- File system latency
- Lock timeout too low

Solutions:

```
# Increase lock timeout
export JOJIAI_LOCK_TIMEOUT=60

# Check file system performance
iostat -x 1

# Monitor concurrent operations
curl http://localhost:8000/metrics | grep concurrent_operations
```

2. Memory Usage Issues

Symptoms: High memory usage, slow performance

Causes:
- Large conversation history
- Memory leaks
- Insufficient cleanup

Solutions:

```
# Check memory usage
memoryctl status

# Cleanup old data
memoryctl cleanup --keep 5

# Monitor memory metrics
curl http://localhost:8000/metrics | grep memory_usage
```

3. WAL File Growth

Symptoms: Large WAL files, disk space issues

Causes:
- Infrequent WAL truncation
- High write volume
- Failed cleanup

Solutions:

```
# Manual WAL cleanup
memoryctl cleanup

# Check WAL size
ls -lh /path/to/memory.wal

# Adjust WAL flush interval
export JOJIAI_WAL_FLUSH_INTERVAL=30
```

4. Backup Failures

Symptoms: Backup errors in logs, missing snapshots

Causes:
- Insufficient disk space
- Permission issues
- Corrupted data

Solutions:

```
# Check disk space
df -h

# Verify permissions
ls -la /path/to/backups

# Manual backup
memoryctl backup --name emergency-backup

# Validate data integrity
memoryctl validate
```

Diagnostic Commands

System Health

```
# Overall system status
memoryctl status

# Detailed health check
curl http://localhost:8000/health | jq

# Check metrics
curl http://localhost:8000/metrics
```

Performance Analysis

```
# Monitor operation latencies
curl http://localhost:8000/metrics | grep duration

# Check error rates
curl http://localhost:8000/metrics | grep errors_total

# Monitor concurrent operations
watch -n 1 'curl -s http://localhost:8000/metrics | grep concurrent'
```

Data Integrity

```
# Validate all data files
memoryctl validate

# Check WAL consistency
python -c "
from jojiai.wal import WriteAheadLog
wal = WriteAheadLog('/path/to/memory.wal')
entries = wal.read_entries()
print(f'WAL entries: {len(entries)}')
"

# Verify backup integrity
memoryctl list-snapshots
```

Maintenance

Regular Tasks

Daily

- Monitor system health and metrics
- Check error logs for issues
- Verify backup completion

Weekly

- Review performance metrics
- Clean up old WAL entries
- Update monitoring dashboards

Monthly

- Analyze usage patterns
- Review and update alert thresholds
- Performance optimization review

Backup Strategy

Automated Backups

- **Frequency**: Every hour
- **Retention**: 24 hourly, 7 daily, 4 weekly, 12 monthly
- **Storage**: Local and remote backup locations

Manual Backups

```
# Before major changes
memoryctl backup --name pre-upgrade-$(date +%Y%m%d-%H%M)

# After configuration changes
memoryctl backup --name post-config-$(date +%Y%m%d-%H%M)
```

Performance Tuning

Memory Optimization

```
# Adjust memory limits
export JOJIAI_MAX_MEMORY_SIZE=209715200  # 200MB

# Optimize dialogue retention
export JOJIAI_MAX_DIALOGUES=500

# Tune decision retention
export JOJIAI_MAX_DECISIONS=100
```

I/O Optimization

```
# Increase WAL flush interval for high-write scenarios
export JOJIAI_WAL_FLUSH_INTERVAL=120

# Adjust backup interval
export JOJIAI_BACKUP_INTERVAL=7200  # 2 hours

# Optimize lock timeout
export JOJIAI_LOCK_TIMEOUT=45
```

Security Considerations

File Permissions

```
# Set secure permissions
chmod 750 /path/to/memory
chmod 640 /path/to/memory/*.json
chmod 600 /path/to/memory.wal
```

Network Security

- Restrict metrics endpoint access
- Use TLS for remote monitoring
- Implement authentication for admin endpoints

Data Protection

- Encrypt sensitive data at rest
- Secure backup storage
- Regular security audits

Recovery Procedures

Data Recovery

From WAL

```
# Recover from WAL entries
python -c "
from jojiai.backup import BackupManager
mgr = BackupManager('/path/to/data', '/path/to/backups', '/path/to/wal')
recovered = mgr.recover_from_wal()
print(f'Recovered {recovered} entries')
"
```

From Backup

```
# List available backups
memoryctl list-snapshots

# Restore specific backup
memoryctl restore snapshot_20230908_120000 --confirm

# Restore to different location
memoryctl restore snapshot_20230908_120000 --target-path /recovery/path
```

Disaster Recovery

Complete System Recovery

1. **Stop the service**
2. **Restore from latest backup**
3. **Replay WAL entries if needed**
4. **Validate data integrity**
5. **Restart service**
6. **Monitor for issues**

```
# Example recovery script
#!/bin/bash
set -e

echo "Starting disaster recovery..."

# Stop service
systemctl stop jojiai

# Restore from backup
memoryctl restore latest_backup --confirm

# Validate integrity
memoryctl validate

# Start service
systemctl start jojiai

# Check health
sleep 10
curl http://localhost:8000/health

echo "Recovery completed successfully"
```

Support

Getting Help

- **Documentation**: GitHub Wiki (https://github.com/gtsurkav-sudo/JOJIAI/wiki)
- **Issues**: GitHub Issues (https://github.com/gtsurkav-sudo/JOJIAI/issues)
- **Discussions**: GitHub Discussions (https://github.com/gtsurkav-sudo/JOJIAI/discussions)

Reporting Issues

When reporting issues, include:

1. System information (`memoryctl status`)
2. Error logs (last 100 lines)
3. Configuration details
4. Steps to reproduce
5. Expected vs actual behavior

Contributing

See CONTRIBUTING.md (CONTRIBUTING.md) for development guidelines.